FH Schmalkalden
Fak. Elektrotechnik

Einführung in Unix

Inhaltsverzeichnis

1. Das Betriebssystem UNIX

Der Begriff Betriebssystem wird durch DIN 44300 folgendermaßen definiert:

> ... diejenigen Programme eines digitalen Rechnersystems, die zusammen mit den Eigenschaften der Rechenanlage die Basis der möglichen Betriebsarten des digitalen Rechensystems bilden und insbesondere die Abwicklung von Programmen steuern und überwachen.

Die wichtigste Aufgabe des Betriebssystemes besteht darin, die zur Verfügung stehenden Ressourcen wie CPU-Zeit, Speicherplatz und Peripheriegeräte (z.B. Festplatten, Bildschirme, Tastaturen, Drucker...) zu verwalten.

UNIX ist multitasking- und multiuserfähig. Multitasking bedeutet, dass mehrere Prozesse quasi parallel laufen. Dabei führt die CPU zu jedem bestimmten Zeitpunkt zwar nur jeweils einen Prozess aus, es wird jedoch so schnell zwischen den verschiedenen Prozessen hin- und hergewechselt, dass der Eindruck der Gleichzeitigkeit entsteht. Multiuser-Betrieb bedeutet, dass mehrere Nutzer gleichzeitig das System benutzen können. Dabei können entweder mehrere Terminals an den Computer angeschlossen sein oder die Nutzer sind über andere Medien wie beispielsweise ein LAN am Computer angemeldet.

2. Die Nutzer unter UNIX

UNIX-Systeme sind für den Betrieb mit mehreren Benutzern ausgelegt. Daraus ergibt sich zwingend die Notwendigkeit, die einzelnen Benutzer voneinander zu unterscheiden.

Jeder Nutzer wird am System durch eine eindeutige Kennummer identifiziert, diese wird als „USER-ID“ (UID) bezeichnet. Jeder Kennummer ist dabei (zumindest für die normalen Nutzer) ein Login-Name zugeordnet, mit dem Sie sich am System anmelden.

Zur Authentifizierung erhält jeder Nutzer ein Passwort. Dieses sollten Sie geheim halten, damit Ihr Benutzer-Account nicht missbraucht werden kann.

Jeder (normale) Nutzer hat ein Home-Verzeichnis, d.h. ein speziell für ihn geschaffenes Verzeichnis, in dem er seine Daten ablegen kann.

Jeder Nutzer ist Mitglied in mindestens einer Gruppe (der sog. primären Gruppe) und kann darüber hinaus Mitglied weiterer Gruppen sein.

Bevor Sie ein UNIX-System nutzen können, muss vom Administrator ein Benutzer-Account für Sie eingerichtet werden. Sie erhalten dann einen Login-Namen und ein Passwort.

3. Die Anmeldung am Rechner

Die meisten modernen UNIX-Systeme bieten Ihnen die Möglichkeit, sich entweder an einer textorientierten Konsole anzumelden oder eine graphische Nutzeroberfläche zu verwenden. Nach Möglichkeit sollten Sie sich an der graphischen Nutzeroberfläche anmelden.

Bei der Anmeldung werden Sie zunächst aufgefordert, Ihren Login-Namen einzugeben. Nachdem Sie diesen mit ENTER bestätigt haben, werden Sie nach Ihrem Passwort gefragt.
Hinweis: Bei der Passwort-Eingabe in einer Textkonsolen-Anmeldung werden Ihre Eingaben nicht angezeigt (auch nicht durch Sternchen).

Nach der Anmeldung wird die graphische Nutzeroberfläche gestartet. Wenn Sie sich an einer Text-Konsole angemeldet haben, wird eine Shell gestartet. Wenn Sie in der graphischen Nutzeroberfläche ein Terminal-Fenster öffnen, wird ebenfalls eine Shell gestartet.

Hinweis: Vergessen Sie nach getaner Arbeit nicht, sich vom Rechner wieder abzumelden!
In einer Text-Konsole geben Sie hierzu

```
logout
```

ein, bei der Arbeit mit graphischen Nutzeroberflächen steht meist ein Button „Exit“ oder „Logout“ zur Verfügung.

4. Die Arbeit mit der Shell

Eine Shell (ein Kommandointerpreter) ist ein Programm, das Ihre Eingaben entgegennimmt, verarbeitet und entsprechende Aktionen veranlasst.

Auf den meisten UNIX-Systemen haben Sie die Auswahl zwischen verschiedenen Shells.
Der X/OPEN-Standard legt fest, dass auf standardkonformen UNIX-Varianten immer die Bourne-Shell (sh) vorhanden ist.
Darüber hinaus wird auf den meisten modernen UNIX-Systemen die C-Shell (csh) mitgeliefert.
Auf UNIX-ähnlichen Open-Source-Betriebssystemen finden Sie meist Derivate wie die Korn-Shell (ksh) und die Turbo-C-Shell (tcsh), die gegenüber sh und csh eine erweiterte Funktionalität bereitstellen.

Die Bourne-Shell und ihre Derivate werden meist für Scripte verwendet, z.B. um beim Systemstart bestimmte Dienste zu starten.
Die C-Shell und ihre Derivate werden häufig als Login-Shells für Nutzer verwendet. Diese Abgrenzung ist allerdings nicht zwingend.

An unserer Fakultät ist standardmäßig für alle Nutzer die C-Shell als Login-Shell eingerichtet, dies kann auf Anfrage des Nutzers geändert werden.

5. Dateien und Verzeichnisse

Dateien sind unter UNIX - wie auch unter anderen Betriebssystemen - eine Ansammlung von Bytes.
Die wichtigsten Dateitypen (für normale Nutzer) sind:

- reguläre Dateien (normale Dateien)
- Directories (Verzeichnisse)
- Spezial-Dateien (special files, dienen zum Ansprechen von Peripheriegeräten)

Verzeichnisse (directories) sind Sammelplätze für Dateien. Jeder Nutzer hat ein Home-Verzeichnis (sein persönliches Verzeichnis), in diesem kann er weitere Unterverzeichnisse anlegen, um die Daten übersichtlich zu organisieren. Auf UNIX-Systemen ist meist eine Hierarchie wie in Bild 1 auf Seite 9 zu finden.

Der Ursprung des Dateisystems ist das sog. root-Verzeichnis, das durch einen Slash „/“ dargestellt wird. Dieses kann Unterverzeichnisse (wie z.B. etc, bin...) enthalten, die ihrerseits wieder Unterverzeichnisse enthalten können...

In Pfadnamen werden die Namen der Unterverzeichnisse durch den Slash (/) voneinander getrennt. Ein Pfadname gibt die genaue Position einer Datei innerhalb des Verzeichnisbaumes an, dies kann entweder absolut (von der Wurzel ausgehend) oder relativ (vom aktuellen Verzeichnis ausgehend) geschehen. Absolute Pfadnamen beginnen mit einem Slash, relative Pfadnamen mit einem anderen Zeichen. Ein Punkt (.) bezeichnet das aktuelle Verzeichnis, zwei Punkte (..) bezeichnen das übergeordnete Verzeichnis.

Zu beachten ist, dass bei Dateinamen unter UNIX zwischen Groß- und Kleinschreibung unterschieden wird.

Sollen bei Kommandos mehrere Dateien angegeben werden, deren Name einem bestimmten Muster folgt, kann dasWildcard-Konzept (manchmal auch Joker-Konzept genannt) verwendet werden.
Sollen beispielsweise die Dateien ueb1.c, ueb2.c und ueb3.c gelöscht werden, so kann anstelle von

```
rm ueb1.c ueb2.c ueb3.c
```

vereinfachend

```
rm ueb*.c
```

geschrieben werden.
Folgende Wildcard-Zeichen bzw. -Konstruktionen können genutzt werden:

Zeichen	Bedeutung
*	beliebige, 0-14 Zeichen lange Zeichenkette
?	exakt ein Zeichen
[xyz]	genau eines der Zeichen x, y oder z
[a-f]	genau ein Zeichen aus dem Bereich a-f

Es können mehrere Wildcards in einem Dateinamen genutzt werden. Der Dateiname mit Wildcards wird dann durch eine Liste mit allen Dateinamen ersetzt, die auf das Muster passen.

Wäre beim Einsatz des obigen Beispieles eine Datei ueb4.c vorhanden, so würde auch diese mit gelöscht.

Werden Wildcards in Löschbefehlen eingesetzt, ist äußerste Vorsicht geboten.

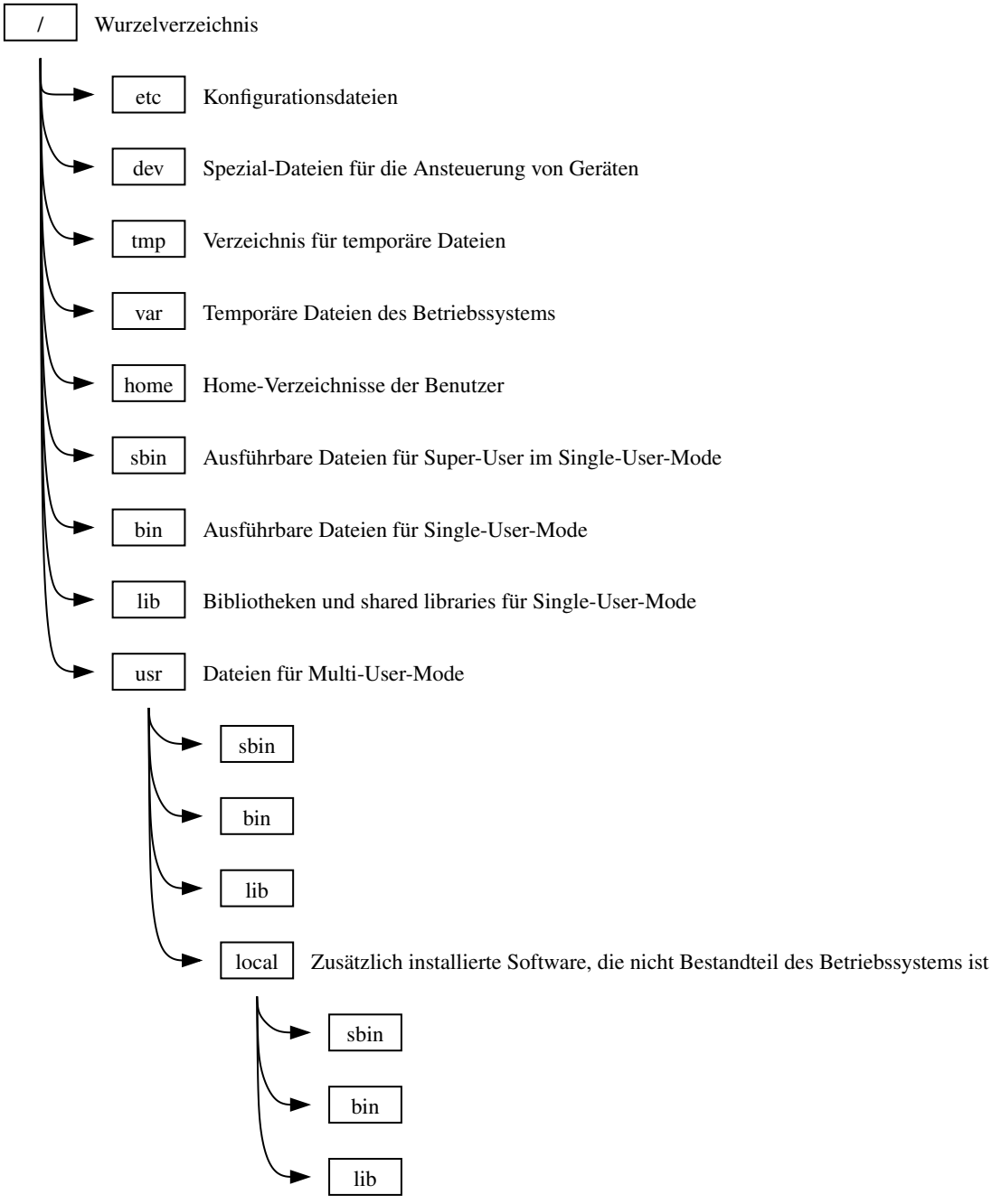

Abbildung 1: Typische Verzeichnis-Struktur auf Unix-Systemen

6. Zugriffsrechte auf Dateien

Unter UNIX sind die Zugriffsrechte (permissions) auf Dateien gestaffelt für

- Eigentümer
- Gruppe des Eigentümers
- alle anderen Nutzer

Jeder dieser drei Personenkreise kann dabei für eine Datei Leserecht (r - read), Schreibrecht (w - write) bzw. Ausführungsrecht (x - execute) besitzen.

Ein Ausführungsrecht für Dateien bedeutet dabei, dass das entsprechende Programm bzw. Script gestartet werden kann. Ein Ausführungsrecht auf Verzeichnisse bedeutet, dass in das Verzeichnis hineingewechselt werden kann.

Da die Rechte für jede Benutzerklasse 3 Bits in Anspruch nehmen, werden die Berechtigungen einer Benutzerklasse durch eine Oktalzahl dargestellt. In dieser Oktalzahl symbolisiert das Bit 2 das Leserecht, Bit 1 das Schreibrecht und Bit 0 das Ausführungsrecht.
Die Oktalzahl wird dann durch eine oder-Verknüpfung der Bits gebildet, es ergibt sich somit folgende Tabelle:

Oktalzahl	**Rechte**
0	`- - -`
1	`- - x`
2	`- w -`
3	`- w x`
4	`r - -`
5	`r - x`
6	`r w -`
7	`r w x`

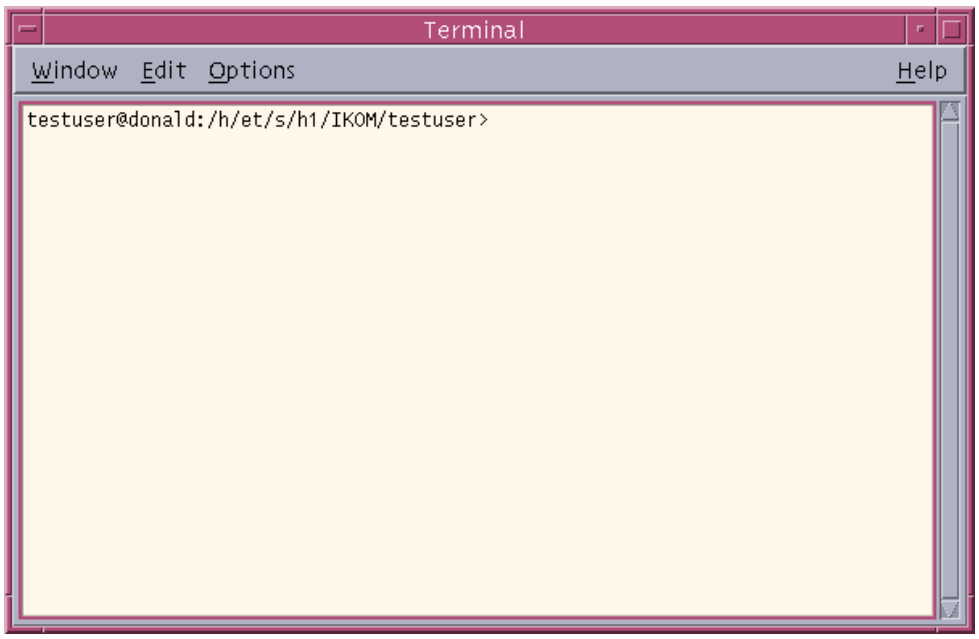

Abbildung 2: Terminal-Fenster

7. Einige wichtige Kommandos

In dieser Einführung können nicht alle UNIX-Kommandos behandelt werden, dazu gibt es derer zu viele.
Ebenso werden nicht alle Optionen besprochen sondern nur die geläufigsten.
Ziel dieses Abschnittes ist es, UNIX-Neulingen zu vermitteln, welches Programm für welchen Zweck eingesetzt werden kann.
Zum Ausprobieren starten Sie zunächst eine Instanz des Programmes dtterm, siehe Abb. 2.

7.1. man - Online-Hilfe

Mit

```
man <Kommando>
```

erhalten Sie Hilfe zum angegebenen Kommando, beispielsweise zeigt

```
man mkdir
```

den Hilfetext für das Programm mkdir.
Die Ausgabe stoppt nach jeweils einer Bildschirmseite. Mit der ENTER-Taste gehen Sie jeweils eine Zeile weiter, mit der Leer-Taste jeweils eine Seite.

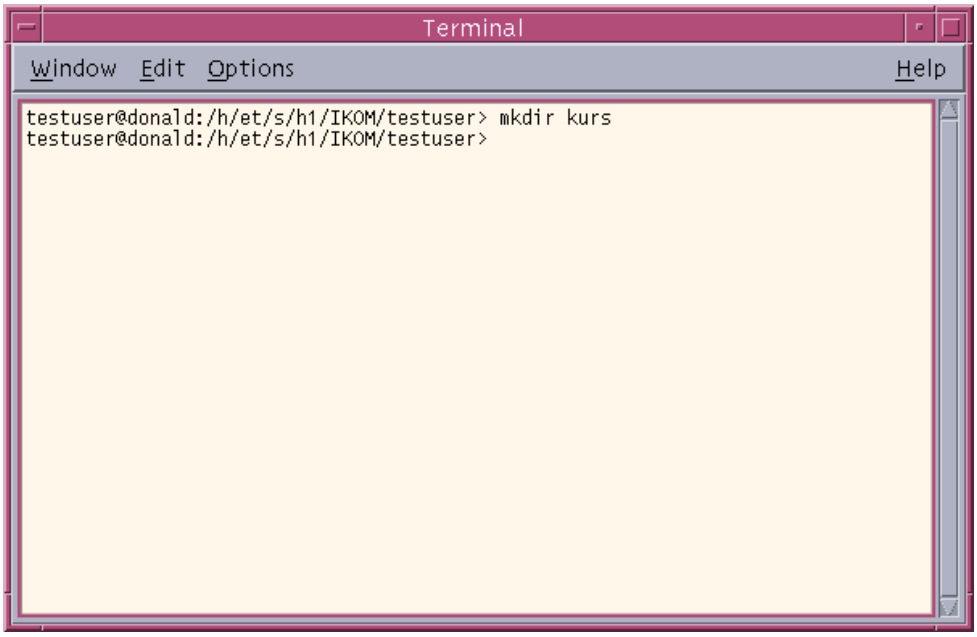

Abbildung 3: Verzeichnis anlegen

7.2. mkdir - Verzeichnis anlegen

Mit dem Kommando „mkdir“ (make directory) kann ein neues Verzeichnis angelegt werden.
Die Syntax des Befehles lautet

```
mkdir <Verzeichnis>
```

Der Name für das neue Verzeichnis kann entweder absolut oder relativ angegeben werden.
Die Verzeichnishierarchie bis zum Elternverzeichnis des neu anzulegenden Verzeichnisses muss bereits existieren.
Abb. 3 zeigt, wie im aktuellen Verzeichnis ein Unterverzeichnis „kurs“ angelegt wird.
Mit

```
mkdir -p <Verzeichnis>
```

wird bei Bedarf die Verzeichnishierarchie automatisch mit angelegt, falls das Elternverzeichnis des neuen Verzeichnisses noch nicht existiert.

Abbildung 4: Verzeichnis wechseln

7.3. cd - Aktuelles Verzeichnis wechseln

Mit dem Befehl „cd“ (change directory) wechseln Sie das aktuelle Arbeitsverzeichnis, die Befehlssyntax lautet

```
cd <Verzeichnis>
```

bzw.

```
chdir <Verzeichnis>
```

siehe Abb. 4. Wird kein Verzeichnisname angegeben, wird in das Home-Verzeichnis gewechselt.

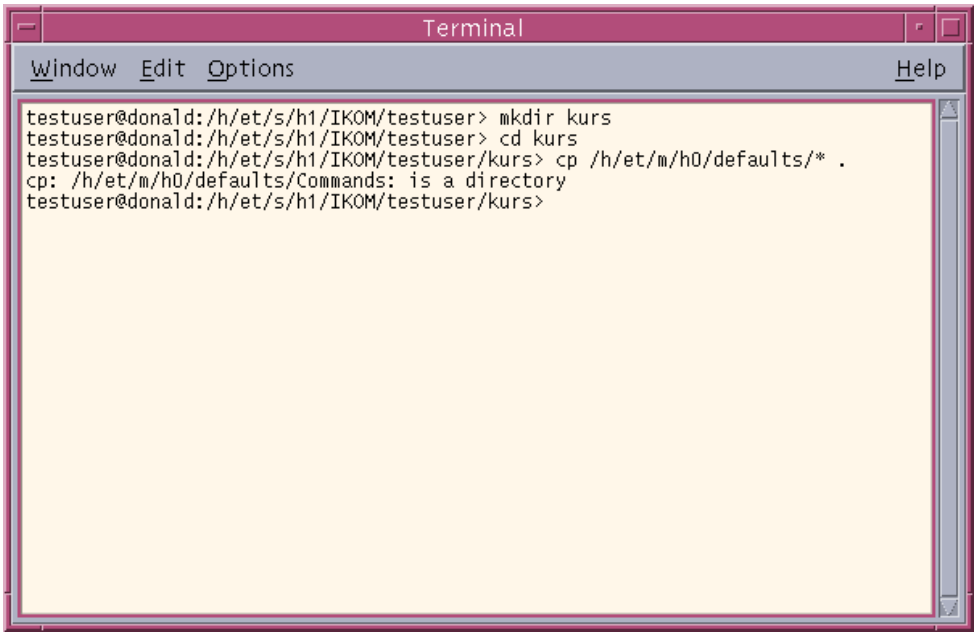

Abbildung 5: Dateien kopieren

7.4. cp - Dateien kopieren

Mit dem Befehl „cp“ (copy) werden Dateien und Verzeichnisse kopiert.
Die Befehlssyntax ist

```
cp <Quelldatei(en)> <Ziel>
```

siehe Abb. 5. Wurden bei den Quelldateien Verzeichnisse angegeben, so wird deren Inhalt standardmäßig nicht mit kopiert, es wird stattdessen eine Warnung ausgegeben. Sollen Unterverzeichnisse und deren Inhalte mit kopiert werden, muss die Option „-r“ angegeben werden, also

```
cp -r <Quelldatei(en)> <Ziel>
```

siehe Abb. 6 auf der nächsten Seite.

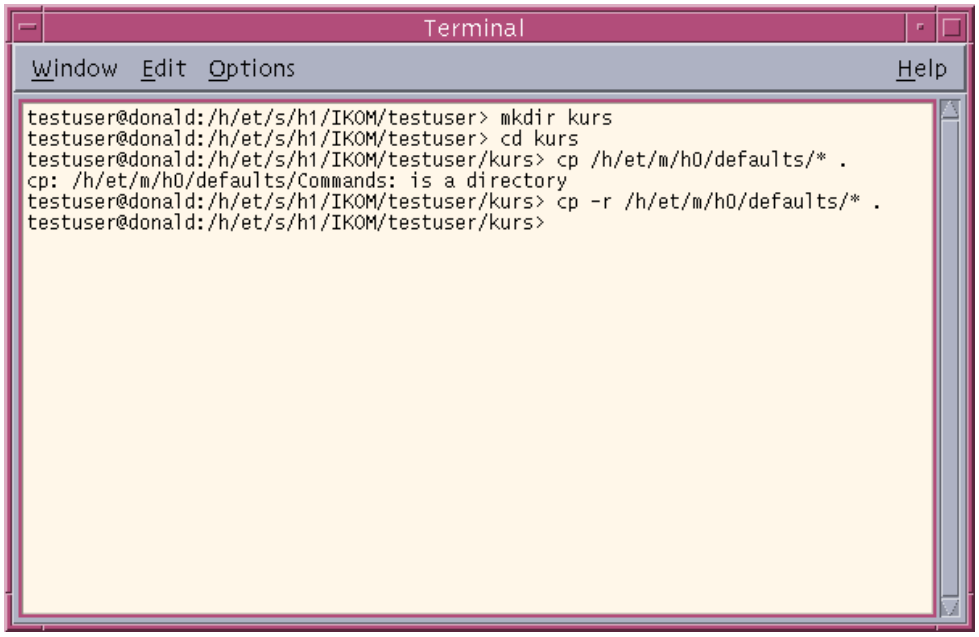

Abbildung 6: Kopieren mit Unterverzeichnissen

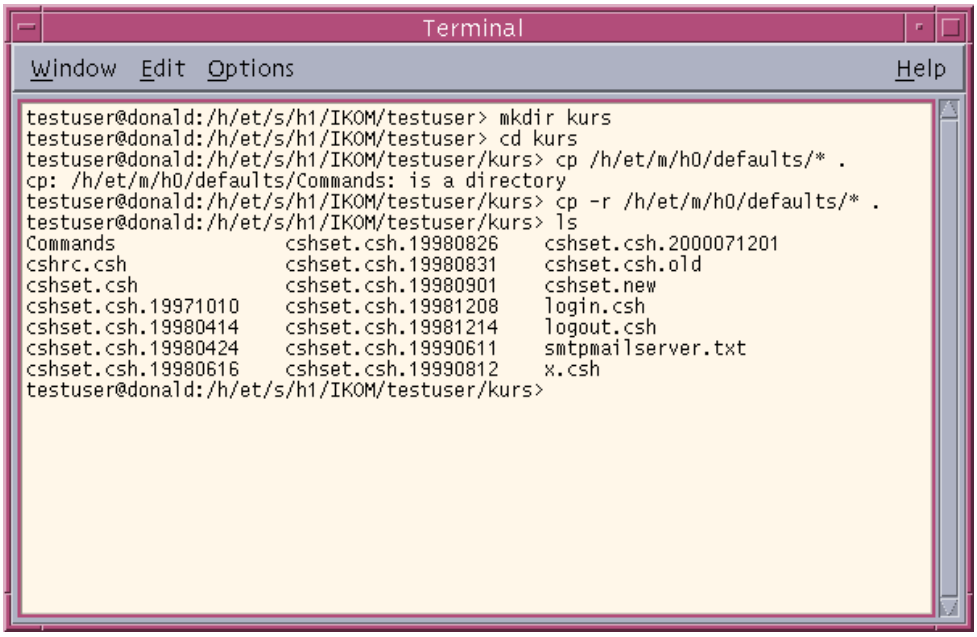

Abbildung 7: Verzeichnisinhalt anzeigen

7.5. ls - Dateien und Verzeichnisse anzeigen

Der Befehl „ls“ (list) dient dazu, Informationen über Dateien und Verzeichnisse anzuzeigen. Mit

```
ls <Datei(en)>
```

werden Informationen über eine Datei bzw. mehrere Dateien ausgegeben, mit

```
ls <Verzeichnis(se)>
```

werden Informationen über (fast) alle Dateien in dem Verzeichnis bzw. den Verzeichnissen ausgegeben.
Mit

```
ls
```

werden Informationen über die Dateien im aktuellen Verzeichnis ausgegeben. In der hier gezeigten Kurzform werden nur die Dateinamen aufgelistet.
Mit der Option „-l“ (long) wird das lange Ausgabeformat angefordert, siehe Abb. 8 auf der nächsten Seite. Im langen Ausgabeformat ist die Ausgabe für jede Datei in mehrere Spalten gegliedert.
Die erste Spalte enthält den Dateityp und die Zugriffsrechte.
Ist der erste Buchstabe ein „d“ handelt es sich um ein Verzeichnis, ist es ein „-“, so handelt es sich um eine reguläre Datei.
Nach dem ersten Buchstaben folgen die Zugriffsrechte als rwx-Tripel für Eigentümer,

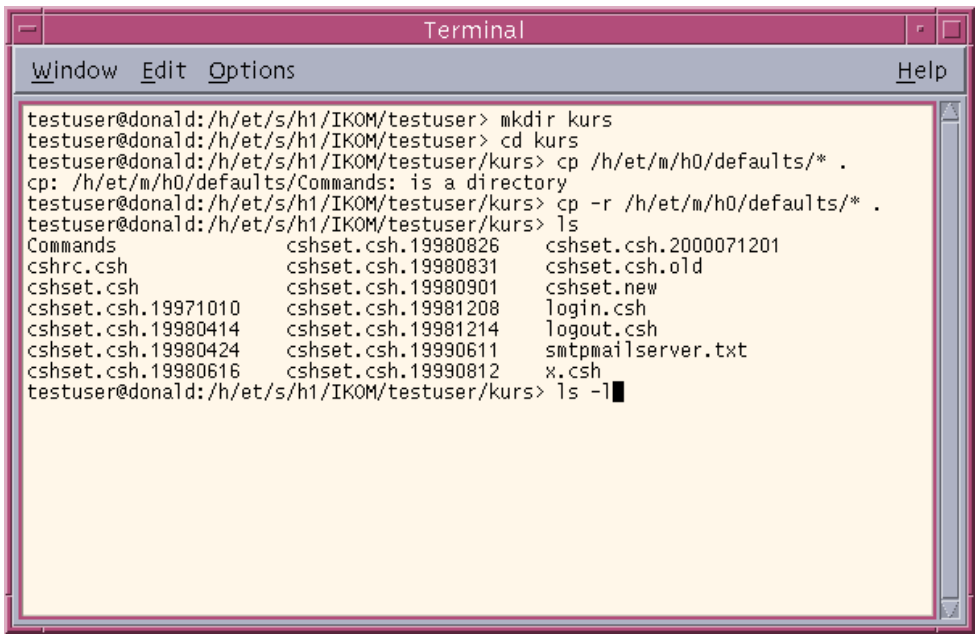

Abbildung 8: Verzeichnisinhalt ausführlich anzeigen

Gruppe und alle anderen Nutzer.
Daran schließt sich die Anzahl der Links auf diese Datei an (wird an späterer Stelle behandelt), der Login-Name des Dateieigentümers, der Name der Gruppe, die Dateigröße in Bytes, der Zeitpunkt der letzten Änderung und zuletzt der Dateiname.

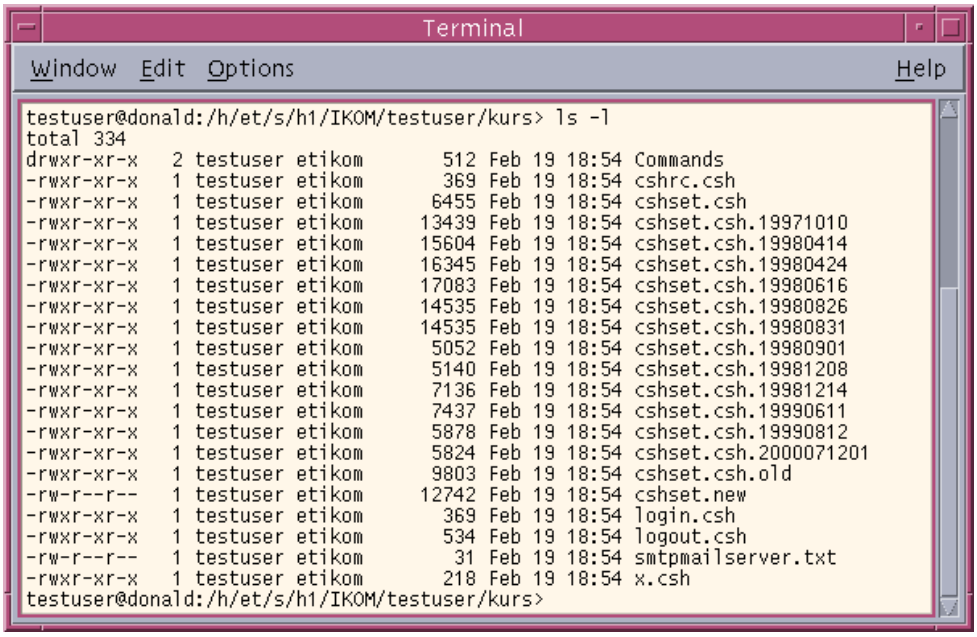

Abbildung 9: Ausführliche Verzeichnisanzeige

7.6. chmod - Zugriffsrechte setzen

Mit dem Programm „chmod“ (change mode) können die Zugriffsrechte auf Dateien und Verzeichnisse geändert werden. Dabei kann man entweder die neuen Zugriffsrechte exakt angeben oder ausgehend von den aktuellen Zugriffsrechten bestimmten Nutzerkreisen zusätzliche Rechte einräumen oder evtl. vorhandene Rechte entziehen.

Um beispielsweise allen anderen Benutzern (o, others) außer dem Eigentümer und dessen Gruppe evtl. vorhandene Lese- und Ausführungsrechte zu entziehen, wird der Befehl

```
chmod o-rx <Datei(en)>
```

eingesetzt, siehe Abb. 10 auf der nächsten Seite für die Ausführung und Abb. 11 auf der nächsten Seite für das Ergebnis.

Um Benutzern zusätzliche Rechte einzuräumen, muss das „-“ durch ein „+“ ersetzt werden.

Für die Benutzerklassen müssen die Kennbuchstaben laut Tabelle eingesetzt werden.

Buchstabe	Bedeutung
u	Eigentümer (user)
g	Gruppe des Eigentümers (group)
o	alle anderen Nutzer (others)

Sollen genau definierte Zugriffsrechte eingestellt werden, verwendet man am besten den Oktalcode.
Soll der Eigentümer Lese- und Schreibrecht auf Dateien erhalten, die Gruppe nur Leserecht und der Rest gar keine Rechte, ergibt sich ein Oktalcode von 640, der Befehl lautet somit

```
chmod 640 <Datei(en)>
```

Die Abbildungen 12 auf Seite 21 und 13 auf Seite 21 zeigen die Ausführung des Befehles und seine Auswirkungen.

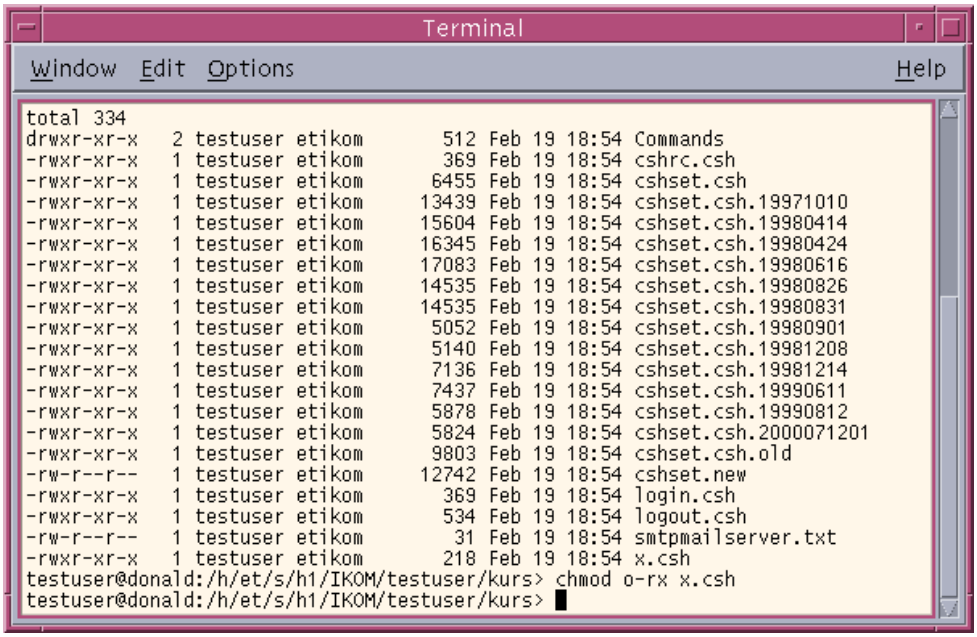

Abbildung 10: Zugriffsrechte entziehen

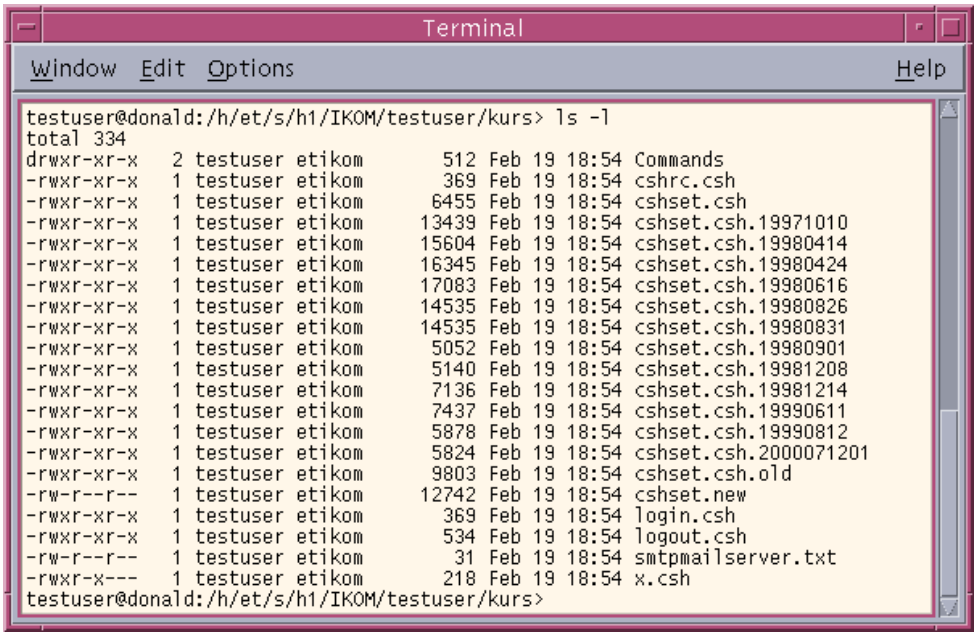

Abbildung 11: Die Zugriffsrechte wurden entzogen

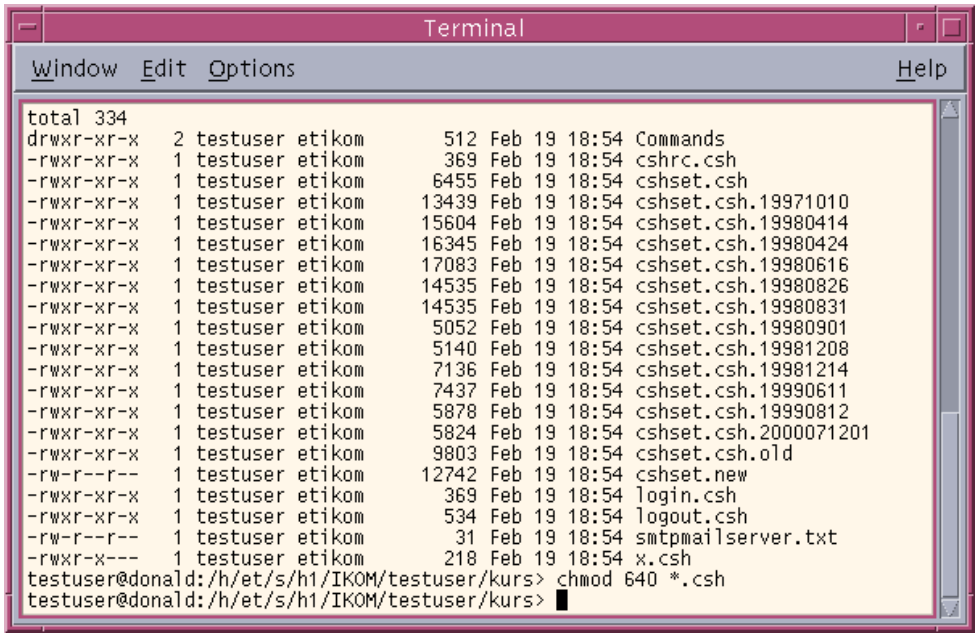

Abbildung 12: Zugriffsrechte exakt festlegen

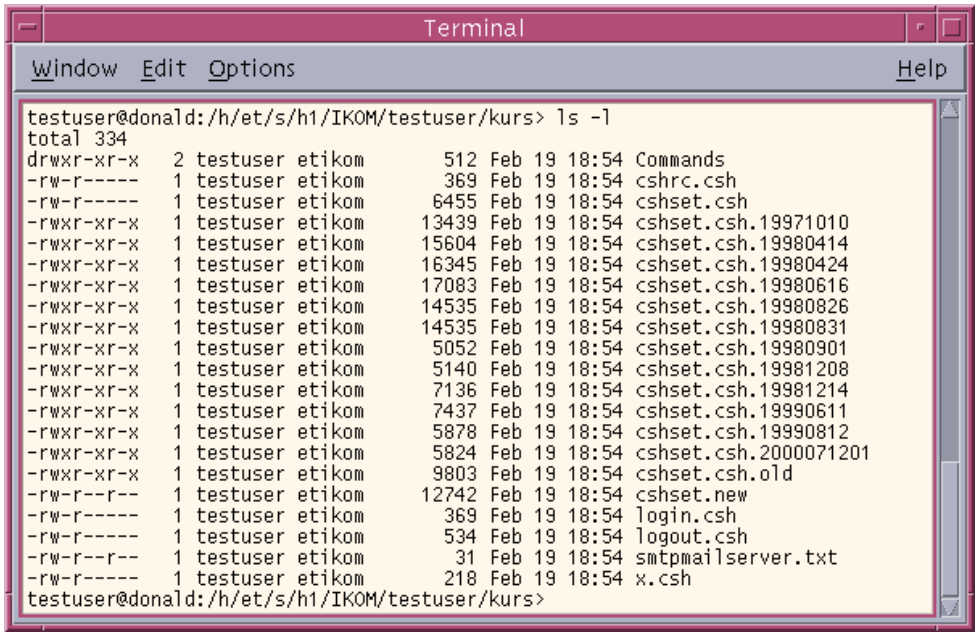

Abbildung 13: Die Zugriffsrechte wurden exakt festgelegt

7.7. umask - Standard-Zugriffsrechte definieren

Mit dem Kommando „umask“, Aufruf

```
umask <Oktalcode>
```

kann festgelegt werden, welche Rechte andere Nutzer auf neu angelegte Dateien *nicht* erhalten sollen.

Beispielsweise wird mit

```
umask 077
```

allen anderen Nutzer jeglicher Zugriff untersagt.
Die auf diese Weise festgelegte Maske gilt in der aktuellen Shell und den Folgeprozessen.
Es wird empfohlen, „umask“ in den Dateien „$HOME/.login“ bzw. „$HOME/.profile“ aufzurufen, siehe „Einführung in csh und sh“[1].

[1] `http://et.fh-schmalkalden.de/labore/wspool/pages/intro/shells.pdf`

7.8. rm - Dateien löschen

Das Programm „rm“ (remove) dient zum Löschen von Dateien, kann aber auch zum Löschen von Verzeichnissen verwendet werden.
Mit

```
rm <Datei(en)>
```

werden die angegebenen Dateien gelöscht, wenn sie reguläre Dateien sind.

Befinden sich unter den angegebenen Dateien Verzeichnisse, so wird eine Fehlermeldung ausgegeben.

Mit dem Schalter „-i“ (interactive) erreicht man, dass vor jedem Löschen einer Datei eine Abfrage erfolgt, ob man die Datei wirklich löschen will.
Mit dem Schalter „-r“ (recursive) veranlasst man, dass auch Verzeichnisse und Unterverzeichnisse samt Inhalt gelöscht werden.
Der Schalter „-f“ (force) schaltet jegliche Rückfragen aus.
Mit

```
rm -fr <Datei(en)>
```

erreicht man also ein unbedingtes Löschen der angegebenen Dateien.

Hinweis: Werden die Optionen „-fr“ genutzt, sollte auf den Einsatz von Wildcards verzichtet werden. Tippfehler wie z.B.

```
rm -fr * .o
```

anstelle von

```
rm -fr *.o
```

haben fatale Folgen, im Beispiel werden alle Dateien anstelle aller *.o-Dateien gelöscht.

7.9. vi - Dateien bearbeiten

Das Programm „vi“ (visual editor) ist ein Texteditor. Eine ausführliche Einführung finden Sie in „Einführung in vi“[2], an dieser Stelle wird nur eine ganz kurze Einführung gegeben.
Der vi verfügt über verschiedene Arbeitsmodi, u.a.

- Auftextmodus.
  Im Auftextmodus können Sie im Textpuffer navigieren (z.B. mit den Cursortasten).

- Eingabemodus. Im Eingabemodus werden die Zeichen, die Sie mit der Tastatur eingeben, in den Textpuffer eingefügt.
  Um vom Auftextmodus in den Eingabemodus zu gelangen, können Sie folgende Tasten benutzen:

Zeichen	Bedeutung
a	Anhängen nach der aktuellen Cursor-Position
i	Einfügen vor der aktuellen Cursor-Position
A	Anhängen nach der aktuellen Zeile
I	Einfügen vor der aktuellen Zeile
o	Einfügen in einer neuen Zeile nach der aktuellen Zeile

  Um vom Eingabemodus wieder in den Auftextmodus zurückzugelangen, drücken Sie die ESC-Taste.

Um eine Datei zu speichern, geben Sie im Auftextmodus

```
:wq
```

ein (ohne führende Leerzeichen). Nach Eingabe des Doppelpunktes wechselt der Cursor in die Statuszeile (unterste Zeile) und zeigt den Doppelpunkt an. Sie befinden sich jetzt im Kommandomodus. Die restlichen Zeichen „wq“ veranlassen, dass der Textpuffer in Datei geschrieben wird (w, write) und anschließend der Editor verlassen wird (q, quit). Das Kommando muss mit ENTER abgeschlossen werden.

Wollen Sie die am Textpuffer vorgenommenen Änderungen nicht speichern, geben Sie

```
:q!
```

ein.

[2] `http://et.fh-schmalkalden.de/labore/wspool/pages/intro/vi.pdf`

7.10. echo - Ausgabe tätigen

Das Programm „echo“ mit der Aufruf-Syntax

```
echo <Argument(e)>
```

gibt die Argumente auf die Standardausgabe aus.
Dies kann beispielsweise in Shell-Scripten genutzt werden, um Ausgaben zu tätigen.

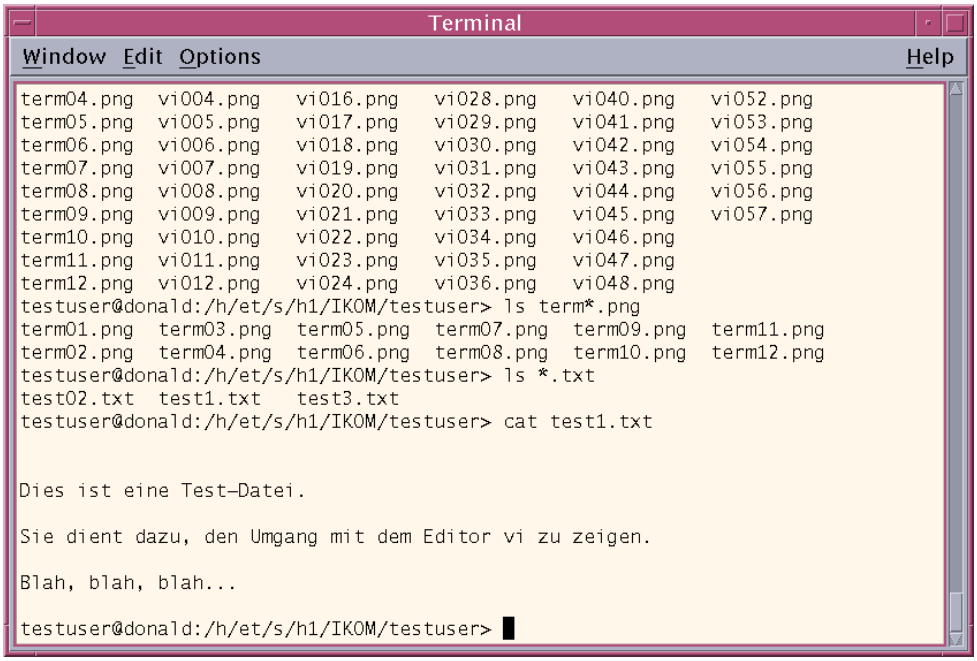

Abbildung 14: Ausgabe von cat

7.11. cat - Datei anzeigen

Das Programm „cat“ (concatenate) gibt eine Datei auf die Standardausgabe aus, der Aufruf lautet

```
cat <Dateiname(n)>
```

Wird kein Dateiname angegeben, wird die Standardeingabe verarbeitet.
Mit der Option „-n“ (number) wird vor jeder Zeile die Zeilennummer ausgegeben (siehe Abb. 15 auf der nächsten Seite).

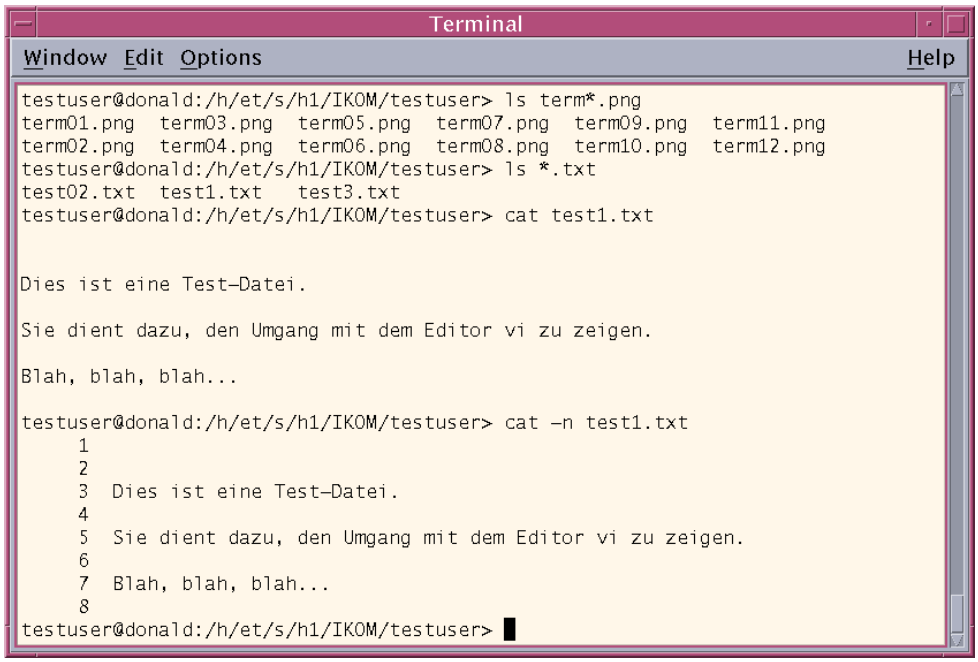

Abbildung 15: Ausgabe von cat -n

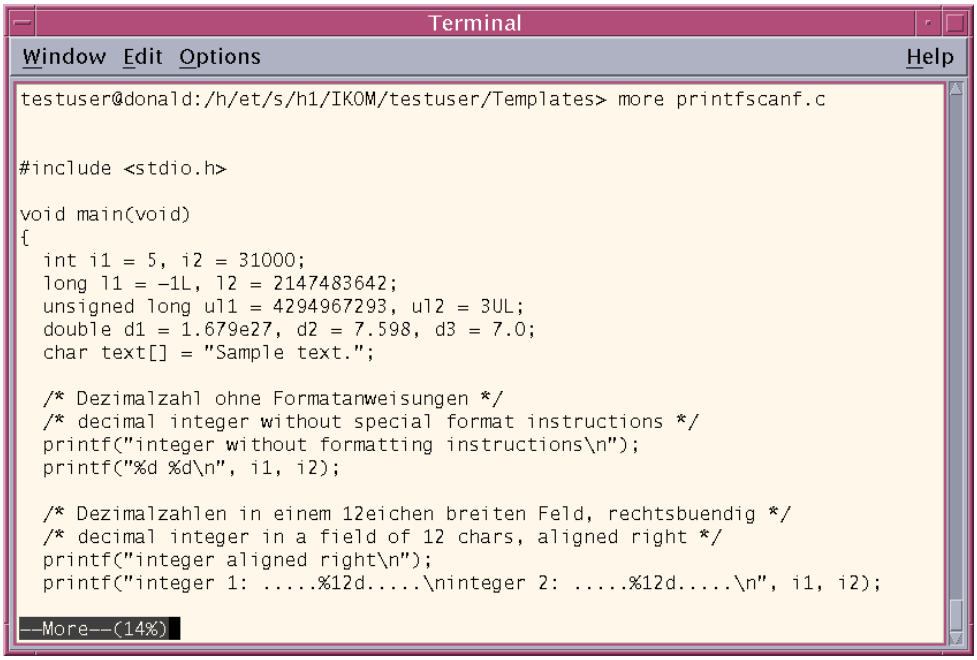

Abbildung 16: Anzeige von more stoppt jeweils nach Bildschirmseite

7.12. more - Datei seitenweise anzeigen

Das Programm „more“ gibt eine Datei seitenweise auf die Standardausgabe aus, der Aufruf lautet

```
more <Dateiname(n)>
```

Wird kein Dateiname angegeben, wird die Standardeingabe verarbeitet.

Nach der ersten Bildschirmseite wird die Ausgabe angehalten (siehe Abb. 16), mit der Leertaste wird die nächste Bildschirmseite angezeigt, mit ENTER läuft die Anzeige eine Zeile weiter.

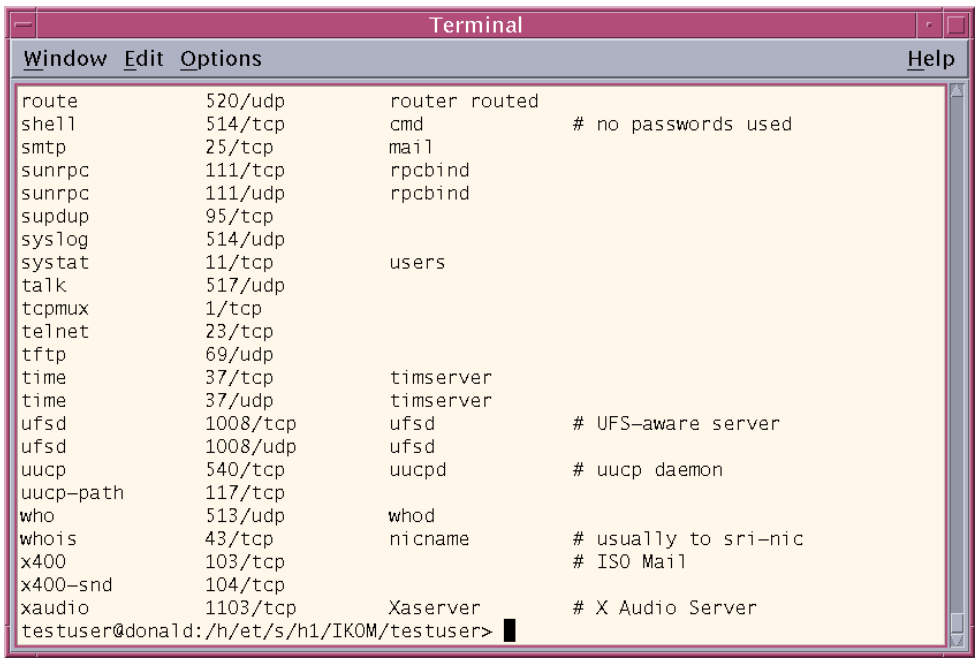

Abbildung 17: Ausgabe von sort

7.13. sort - Zeilenweise sortieren

Sollen die Zeilen einer Datei sortiert angezeigt werden, kann dies mit dem Programm „sort“ mittels

```
sort <Datei(en)>
```

erfolgen. Wird kein Dateiname angegeben, wird die Standardeingabe verarbeitet.
Im Beispiel in Abb. 17 wurde mit

```
sort /etc/services
```

die Liste der Netzwerkdienste (die normalerweise nach Portnummern sortiert ist) alphabetisch geordnet. Einige Optionen für sort sind:

- -b
  ignoriert Leerzeichen und Tabulatoren am Zeilenanfang
- -n
  sortiert nach Zahlenwert
- -r
  Sortierreihenfolge umkehren.

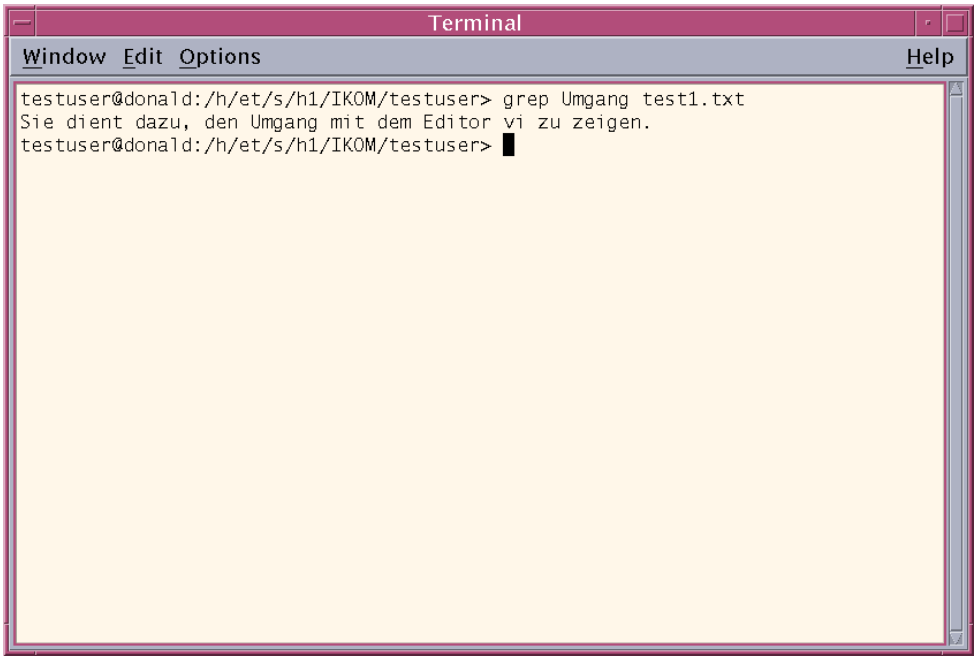

Abbildung 18: Durchsuchen einer Datei mit grep

7.14. grep - Dateiinhalt durchsuchen

Mit

```
grep <Suchmuster> <Datei(en)>
```

kann in der angegebenen Datei bzw. den angegebenen Dateien nach einem Suchmuster gesucht. Wird kein Dateiname angegeben, wird die Standardeingabe verarbeitet.

Die Zeilen, die das Suchmuster enthalten, werden auf die Standardausgabe ausgegeben.
Wird nur ein Dateiname zum Durchsuchen angegeben, werden nur die passenden Zeilen auf die Standardausgabe ausgegeben, siehe Abb. 18.

Werden mehrere Dateien durchsucht, wird vor jeder Textzeile der Dateiname ausgegeben, siehe Abb. 19 auf der nächsten Seite. Es gibt mehrere Varianten des Programme „grep“, die das Suchmuster auf unterschiedliche Weise interpretieren:

- fgrep interpretiert das Suchmuster als exakte Zeichenfolge, nach der gesucht werden soll.
- grep interpretiert das Suchmuster als einfachen regulären Ausdruck (siehe Anhang A auf Seite 53).
- egrep interpretiert das Suchmuster als erweiterten regulären Ausdruck.

Suchmuster können einerseits als exakte Zeichenfolge angegeben werden, andererseits kann vorgegeben werden, dass eine Zeichenfolge in ihrem Aufbau einem be-

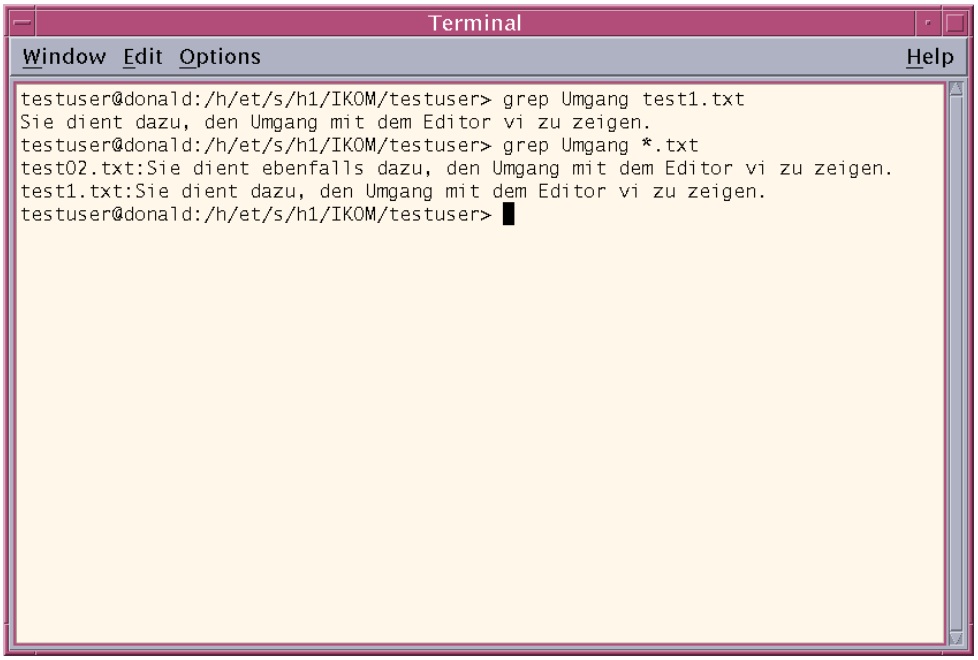

Abbildung 19: Durchsuchen mehrerer Dateien mit grep

stimmten Schema gehorchen soll. Ein derartiges Schema wird alsregulärer Ausdruck bezeichnet.

Bei der Definition regulärer Ausdrücke kommen Sonderzeichen zum Einsatz, die normalerweise von der Shell als Wildcards interpretiert werden. Daher sollte das Suchmuster - wenn es ein regulärer Ausdruck ist - in einfache Anführungszeichen eingeschlossen werden, um den Wildcard-Mechanismus zu unterbinden.

Im Beispiel in Abb. 20 auf der nächsten Seite wird mit

```
grep 'U[a-z]*' *.txt
```

in allen Dateien mit der Endung .txt nach einer Zeichenfolge gesucht, die mit einem großen „U“ beginnt und an die sich Kleinbuchstaben „a“...„z“ ([a-z]) in beliebiger Anzahl (*) - d.h. 0- oder mehrfach - anschließen.
Diese Mustervorgabe trifft sowohl für „Umgang“ als auch für „Usage“ zu.

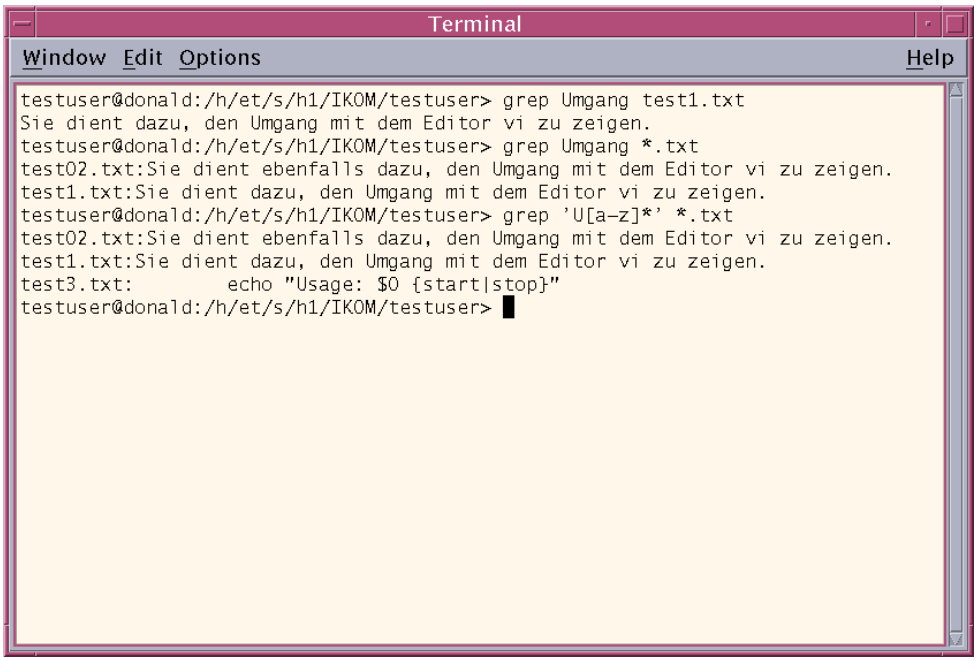

Abbildung 20: Suche mit regulärem Ausdruck

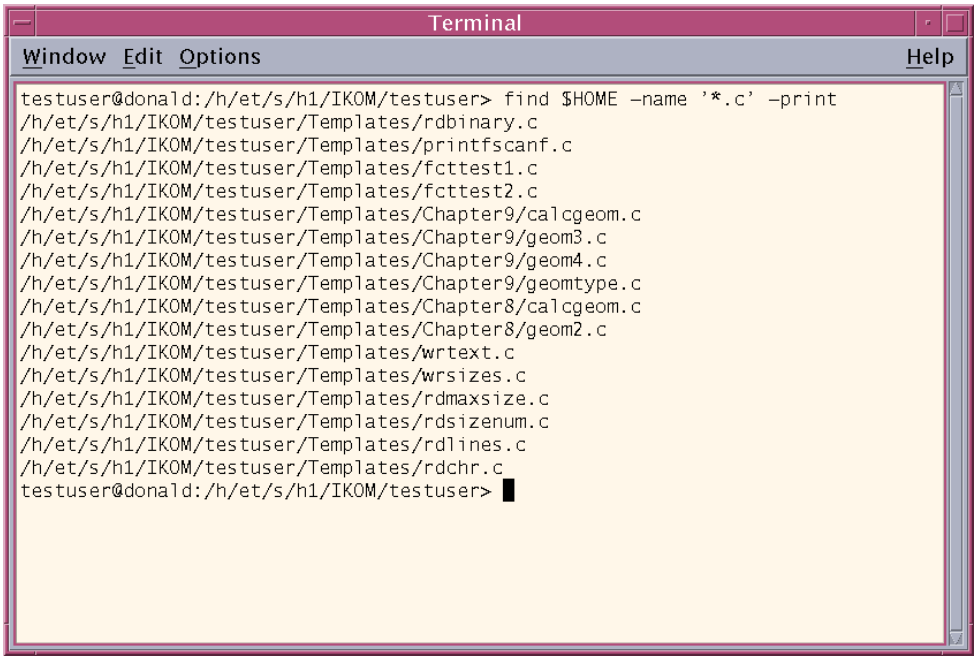

Abbildung 21: Suche nach bestimmten Dateinamen, Ausgabe der Namen

7.15. find - Dateien suchen

Mit dem Programm „find“ kann ein Verzeichnis nach bestimmten Dateien durchsucht werden.
Werden passenden Dateien gefunden, kann der Name der Dateien ausgegeben werden oder ein bestimmter Befehl für jede Datei gestartet werden.
Das Programm wird mit

```
find <Verzeichnis> <Auswahlkriterium> <Aktion>
```

aufgerufen. Beispiel:

```
find $HOME -name '*.c' -print
```

sucht im Home-Verzeichnis alle Dateien mit der Endung .c und gibt deren Namen aus (Abb. 21).

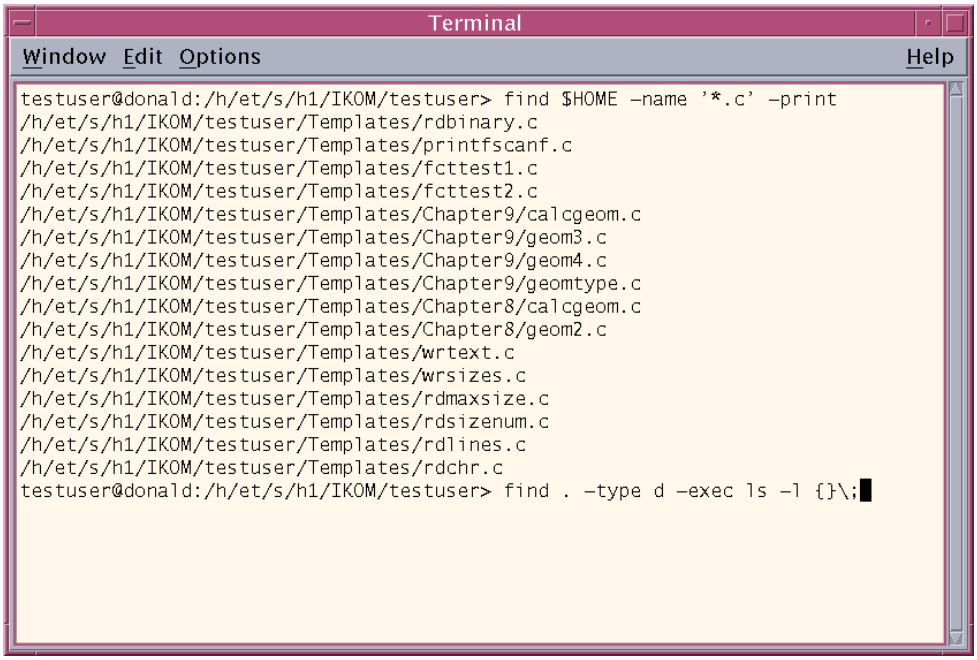

Abbildung 22: Suche nach Dateityp, Ausführung eines Kommandos

Mit

```
find . -type d -exec ls -l {} \;
```

wird das aktuelle Verzeichnis (.) nach Verzeichnissen (-type d) durchsucht, für jede gefundene Datei wird der Befehl

```
ls -l {}
```

ausgeführt (-exec ... \;). Dabei werden die geschweiften Klammern bei jeder einzelnen Datei durch den jeweils aktuellen Dateinamen ersetzt.

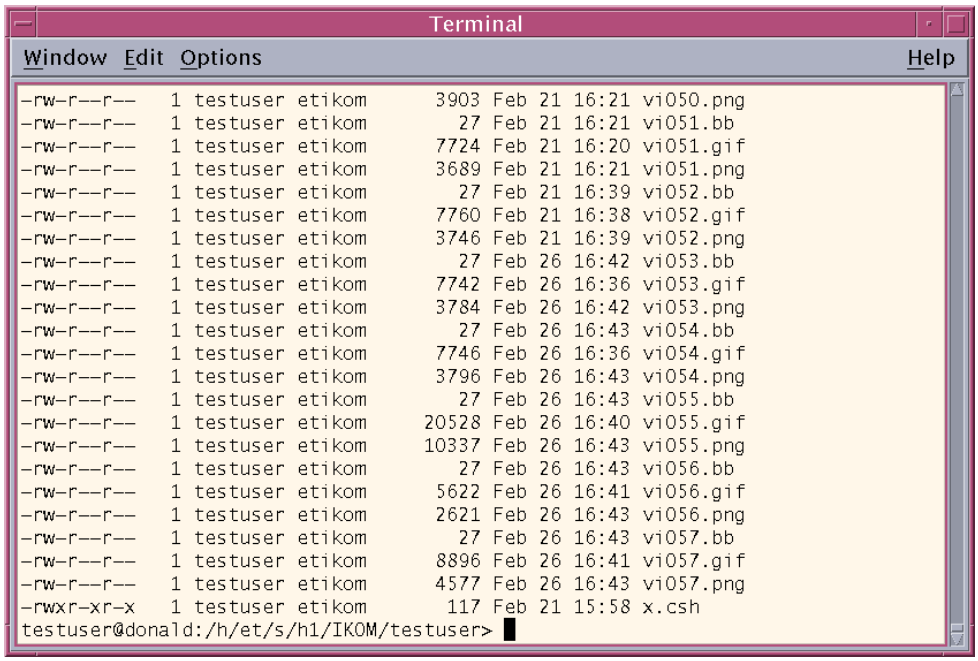

Abbildung 23: Ausgabe des find-Befehls

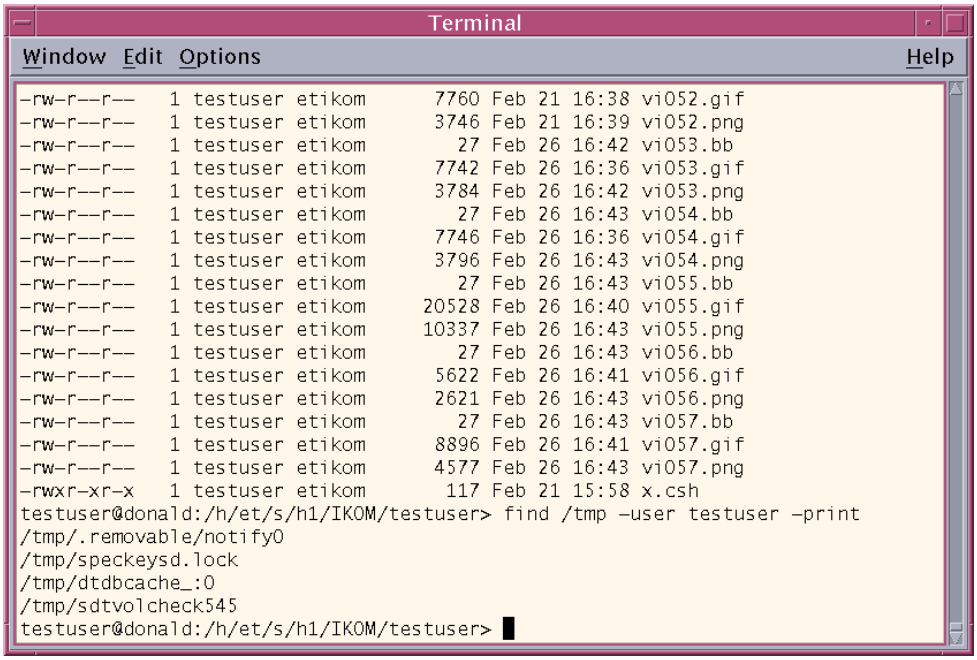

Abbildung 24: Suche nach Eigentümer

Mit

```
find /tmp -user testuser -print
```

werden im Verzeichnis /tmp alle Dateien gesucht, die dem Nutzer testuser gehören und deren Name ausgegeben.

7.16. expr - Berechnung von Ausdrücken

Das Programm „expr“ dient zum Auswerten und Berechnen von mathematischen und logischen Ausdrücken sowie Text-Operationen. Das Ergebnis wird auf die Standardausgabe ausgegeben.
Die Befehls-Syntax lautet

```
expr <Argument...>
```

dabei bezeichnet <*Argument...*> den Ausdruck, der berechnet werden soll.
Folgende Konstruktionen stehen für die Bildung eines Ausdruckes zur Verfügung:

Konstrukt	Bedeutung
Ausdruck | *Ausdruck*	logische oder-Verknüpfung zweier Ausdrücke, der Backslash nimmt dem |-Zeichen die Sonderbedeutung als Pipe.
Ausdruck \& *Ausdruck*	logische und-Verknüpfung zweier Ausdrücke, der Backslash nimmt dem &-Zeichen die Sonderbedeutung des Startens von Hintergrundprozessen
Ausdruck = *Ausdruck*	Integer-Vergleich, wenn beide Operanden ganzzahlig sind, ansonsten String-Vergleich
< > <= >= !=	weitere Vergleichsoperatoren
Ausdruck + *Ausdruck*	Addition zweier Ausdrücke
- * / %	Subtraktion, Mulitiplikation (Backslash nimmt dem Stern die Sonderbedeutung), Division, Modulo-Division
String : BRE	prüft, ob der *String* den einfachen regulären Ausdruck BRE erfüllt.

Je nach verwendeter UNIX-Version sind weitere Optionen verfügbar.
Beispiel:

```
a=‘expr $a ’*’ 2‘
```

verdoppelt den Wert der Shell-Variablen *a* (in der Bourne-Shell).

7.17. test - Test verschiedener Bedingungen

Das Programm „test“, Aufruf

```
test <Testausdruck>
```

testet, ob der angegebene Testausdruck wahr ist.

Im Erfolgsfall wird das Programm mit dem Exit-Status 0 beendet, alle anderen Rückgabewerte deuten darauf hin, dass der Test fehlgeschlagen ist. Es können Tests mit Zeichenketten, Zahlen und Dateinamen durchgeführt werden, Testausdrücke können logisch miteinander verknüpft werden. Testausrücke können folgendermaßen aufgebaut werden:

Konstrukt	Bedeutung
-f *Dateiname*	die Datei existiert und ist eine reguläre Datei
-d *Dateiname*	die Datei existiert und ist ein Verzeichnis
-s *Dateiname*	die Datei existiert und ist größer als 0 Bytes
-r *Dateiname*	die Datei existiert und ist lesbar
-w *Dateiname*	die Datei existiert und ist schreibbar
-x *Dateiname*	die Datei existiert und ist ausführbar
-h *Dateiname*	die Datei existiert und ist ein symbolischer Link
-c *Dateiname*	die Datei existiert und ist ein character special device
-b *Dateiname*	die Datei existiert und ist ein block special file
-p *Dateiname*	die Datei existiert und ist ein named pipe
-u *Dateiname*	die Datei existiert und das Set-UID-Bit ist gesetzt
-g *Dateiname*	die Datei existiert und das Set-GID-Bit ist gesetzt
-k *Dateiname*	die Datei existiert und das Sticky-Bit ist gesetzt
-z *String*	die Länge des Strings ist 0
-n *String*	die Länge des Strings ist größer als 0.
String1 = *String2*	die Strings sind identisch
String1 != *String2*	die Strings sind nicht identisch
String	Es handelt sich nicht um einen leeren String
String1 -eq *String2*	die Strings sind algebraisch gleich
String1 -ne *String2*	die Strings sind nicht algebraisch gleich
String1 -gt *String2*	*String1* ist algebraisch größer als *String2*
String1 -lt *String2*	... kleiner ...
String1 -le *String2*	... kleiner oder gleich ...
! *Ausdruck*	unäre Negation des Ausdrucks
Ausdruck1 -o *Ausdruck2*	logische oder-Verknüpfung der Ausdrücke
Ausdruck2 -a *Ausdruck2*	logische und-Verknüpfung der Ausdrücke
(*Ausdruck*)	Klammerung des Ausdrucks zur Festlegung von Vorrang
-t *Filedeskriptor*	Der Filedeskriptor ist einem Terminal zugeordnet

8. Filter, Pipes und Eingabeumlenkung

Etliche Kommandos unter UNIX sind als Filter einsetzbar, d.h.

- die zu verarbeitenden Daten werden über die Standardeingabe bezogen,
- Ergebnisse werden auf die Standardausgabe ausgegeben und
- eventuelle Fehlermeldungen erfolgen auf die Standardfehlerausgabe.

Normalerweise ist die Standardeingabe mit der Tastatur verknüpft, Standardausgabe und Standardfehlerausgabe sind mit dem Bildschirm verknüpft. Zu einer Pipe können zwei Kommandos mit dem Operator | verknüpft werden. In diesem Fall erhält das zweite Kommando seine Standardeingabe aus der Standardausgabe des ersten Kommandos.
Mit

```
ls -l | sort
```

wird beispielsweise die Ausgabe von

```
ls -l
```

durch

```
sort
```

weiterverarbeitet.

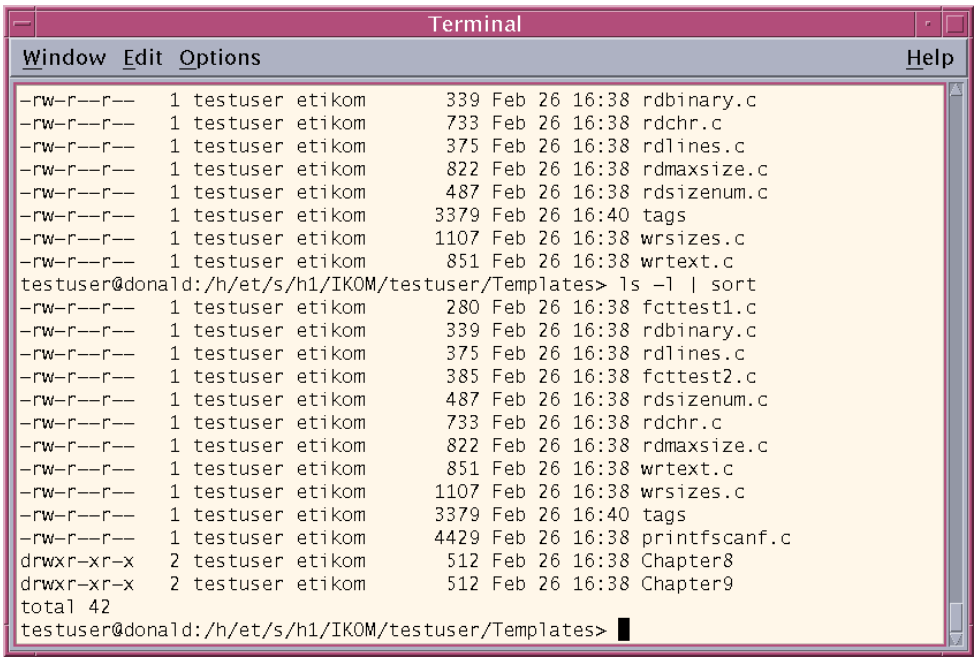

Abbildung 25: Pipe

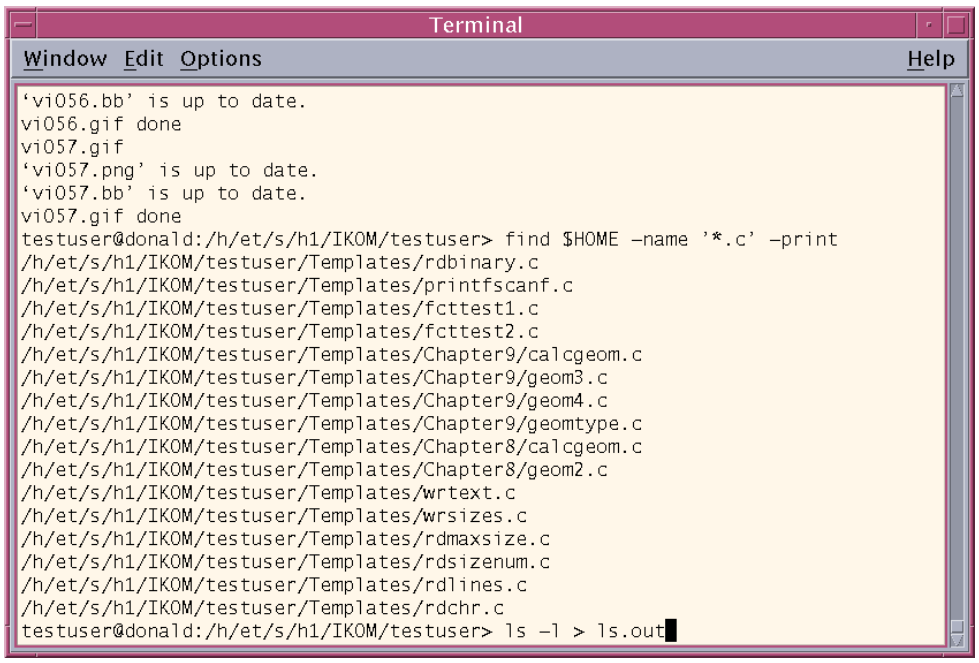

Abbildung 26: Umleitung der Ausgabe

Die Ausgabe eines Programmes kann mit dem Operator „>“

```
<Kommando> > <Datei>
```

in eine Datei umgeleitet werden.
Beispielsweise speichert

```
ls -l > ls.out
```

die Ausgabe des Programmes

```
ls -l
```

in der Datei ls.out. Anstelle die Ausgabe auf dem Bildschirm anzuzeigen, werden die Daten nun in der angegebenen Datei gespeichert, die beispielsweise mit einem Texteditor eingesehen werden kann.

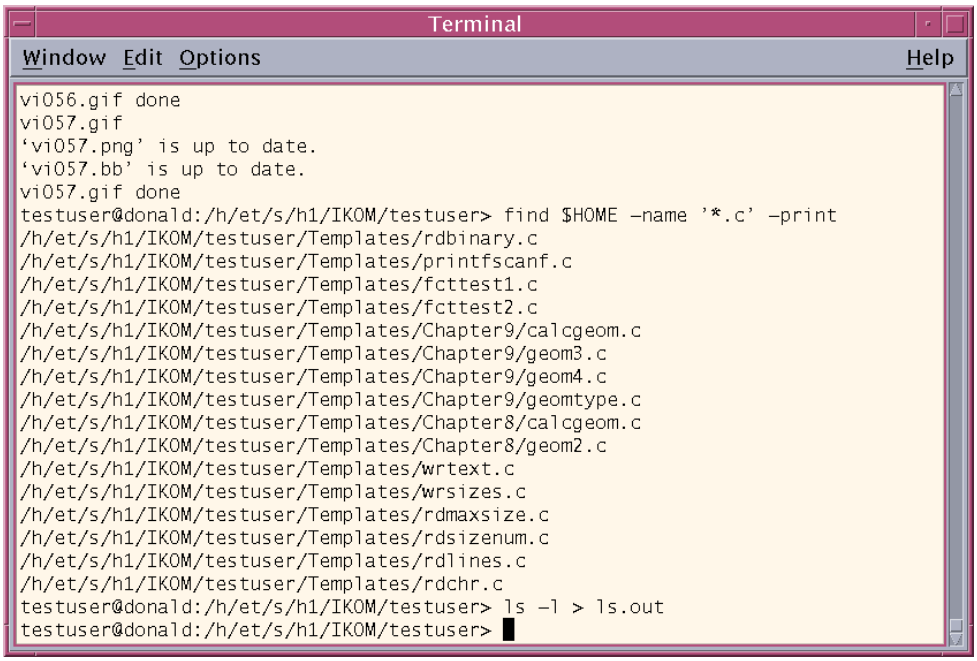

Abbildung 27: Keine Ausgabe auf Bildschirm

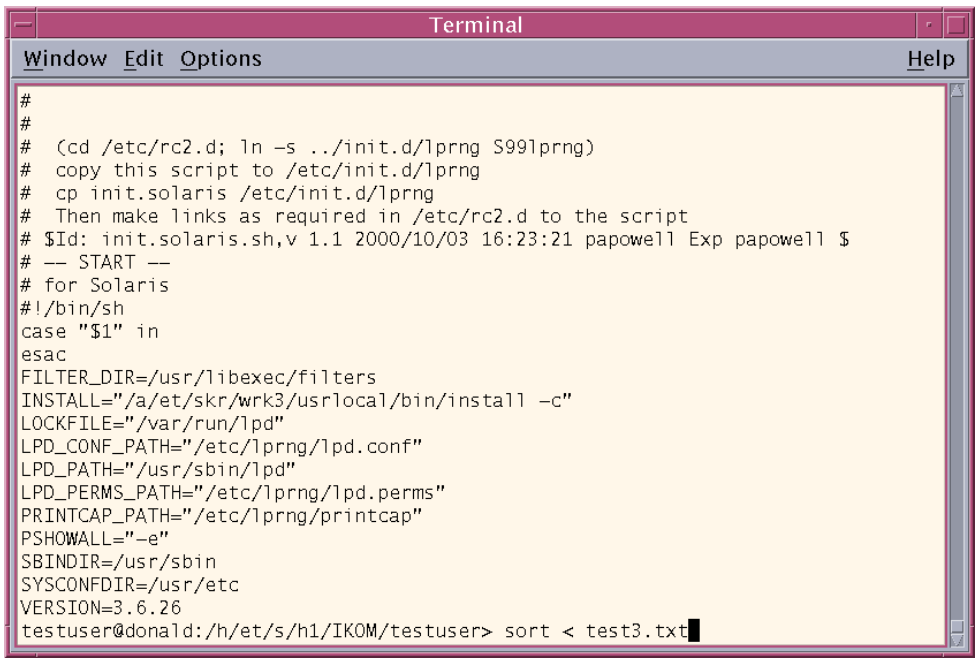

Abbildung 28: Umlenkung der Standardeingabe

Die Eingabe kann ebenfalls umgeleitet werden, d.h. anstelle von Tastatureingaben wird die genannte Datei eingelesen. Hierfür wird auf den Operator „<“ zurückgegriffen mittels

```
<Kommando> < <Datei>
```

Bespielsweise wird mit

```
sort < test3.txt
```

der Inhalt der Datei test3.txt sortiert.

Die Umlenkung der Standardfehlerausgabe ist ebenfalls möglich, die Art und Weise hängt allerdings davon ab, welche Shell verwendet wird.

9. Etwas genauer hingeschaut

9.1. Dateisystem

Für jede Datei speichert das UNIX-Dateisystem sowohl den Inhalt der Datei als auch zusätzliche Informationen über die Datei. Zu diesen zusätzlichen Informationen gehören u.a.:

- Dateityp (reguläre Datei, Verzeichnis, special file, socket, pipe...),
- Zugriffsrechte auf die Datei,
- User-ID und Gruppen-ID des Eigentümers,
- Zeitpunkte von Erzeugung, letzter Modifikation und letztem Zugriff und
- Position der in der Datei enthaltenen Daten auf dem Datenträger.

Diese Zusatzinformation wird in einem sogenannten inode gebündelt.
Ein Teil eines jeden Dateisystemes (Festplattenpartition) wird dabei für die Inode-Tabelle reserviert, der restliche Platz steht dann für die Daten zur Verfügung.

In Verzeichnissen werden für alle im Verzeichnis enthaltenen Dateien die Position des Inodes (als Index in der Inode-Tabelle) und der Dateiname gespeichert.
Es ist möglich - und durchaus nicht ungewöhnlich - dass in verschiedenen Verzeichnissen unter verschiedenen Namen auf denselben Inode und somit auf dieselbe Datei verwiesen wird. D.h. ein und dieselbe Datei kann in verschiedenen Verzeichnissen unter verschiedenen Namen enthalten sein.
Die Anzahl der Links in der Ausgabe von

```
ls -l
```

gibt an, wieviele Verweise auf eine Datei existieren.
Wird eine Datei gelöscht, wird zunächst nur der sich aus dem Dateinamen ergebende Verweis im entsprechenden Verzeichnis entfernt und die Anzahl der Links reduziert. Nur wenn die Anzahl der Links beim Löschen 0 wird, können die Daten gelöscht und der Inode-Eintrag freigegeben werden.
Verzeichnisse haben üblicherweise Link-Zahlen größer als eins, da es im übergeordneten Verzeichnis einen Verweis auf das Verzeichnis gibt, das Verzeichnis unter dem Namen „.“ einen Verweis auf sich selbst enthält und alle direkten Unterverzeichnisse einen Verweis mit dem Namen „..“.

9.2. Dateisystem-Pufferung

Wenn ein Programm Daten schreibt, erfolgt nicht sofort in jedem Fall ein Schreibvorgang auf den Datenträger. Im Normalfall werden die Daten im Hauptspeicher gepuffert, so dass blockweise auf den Datenträger geschrieben werden kann. Auch

Lesevorgänge erfolgen gepuffert. In regelmäßigen Abständen bzw. bei geringer Belastung des Rechners werden die Pufferinhalte auf die Datenträger übertragen.

Ein normales Programm kann das Schreiben der Puffer auf die Datenträger nicht beeinflussen sondern lediglich veranlassen, dass eine Übertragung zwischen den programmeigenen Puffern und den Systempuffern erfolgt.

Die Synchronisation der Systempuffer mit den Datenträgern kann durch Aufruf des Kommandos

```
sync
```

angestoßen werden, dabei wird dem Betriebssystem allerdings lediglich vorgeschlagen, sofort eine Synchronisation durchzuführen.

Beim Aushängen (unmount) von schreibbaren Datenträgern bzw. beim Herunterfahren wird eine letzte Synchronisation vorgenommen und in speziellen, auf dem Datenträger gespeicherten Flags vermerkt, dass alle Änderungen gespeichert wurden.

Fand keine Synchronisation statt (z.B. weil wegen Stromausfall das System nicht heruntergefahren wurde) merkt das System beim nächsten Start anhand des nicht gesetzten Flags, dass möglicherweise Daten verloren gingen.
Es wird ein File-System-Check angestoßen, bei welchem die Integrität des Dateisystemes überprüft wird. Hierbei sollte der Administrator zugegen sein, um anhand der Ausgaben des File-System-Checks über notwendige Maßnahmen (Neuinstallation des Systems, Einspielen von Backups...) entscheiden zu können. Üblicherweise ist das Hoch- und Herunterfahren sowie das Ausschalten von UNIX-Workstations den Systembetreuern (und ausreichend qualifizierten Nutzern) vorbehalten.

Da bei Stromausfällen bzw. Netzschwankungen Datenverluste auftreten können, sollte eine USV (unterbrechungsfreie Stromversorgung) zum Einsatz kommen. Diese ist in der Lage, einen gewissen Zeitraum zu überbrücken und dem System den Stromausfall zu signalisieren. Damit kann ein geordnetes Herunterfahren des Systems erreicht werden.

9.3. Prozesse

9.3.1. Übersicht

Jede Rechneraktivität im Betriebssystem UNIX vollzieht sich in einem Prozess. Als Prozess wird eine Instanz eines laufenden Programmes bezeichnet.
Wird beispielsweise in mehreren Terminalfenstern jeweils das Kommando

```
ls
```

eingegeben, so wird zwar (normalerweise) immer dasselbe Programm ls aus derselben Datei (meist /usr/bin/ls) gestartet, es sind jedoch mehrere Prozesse aktiv, die jeweils ihre eigene Standardeingabe, Standardausgabe und Standardfehlerausgabe besitzen. Weiterhin verfügt jeder Prozess über Informationen über sein aktuelles Verzeichnis, Unterbrechungsbehandlung und Umgebungsvariablen.

9.3.2. Prozesse anzeigen

Mit

```
ps
```

können Sie sich laufende Prozesse anzeigen lassen. Allerdings werden hier nur ausgewählte Prozesse angezeigt.
Mit

```
ps -ef
```

wird die Anzeige wesentlich erweitert. Dabei sorgt „-e“ (every) dafür, dass alle Prozesse angezeigt werden und „-f“ (full) für die Ausgabe aller erhältlichen Informationen, siehe Abb. 29 auf der nächsten Seite).
Die Ausgabe erhält für jeden laufenden Prozess folgende Angaben:

- UID
  Der Nutzername (bzw. die User-ID) unter der der Prozess läuft.
- PID
  Die Prozess-ID. Jedem Prozess ist eine eindeutige Nummer zugeordnet.
- PPID
  Die parent-process-id, PID des Eltern-Prozesses (d.h. des Prozesses, der den jeweiligen Prozess gestartet hat).
- CPU
  Der prozentuale Anteil an CPU-Zeit, die der Prozess einnimmt.
- STIME
  Der Startzeitpunkt.

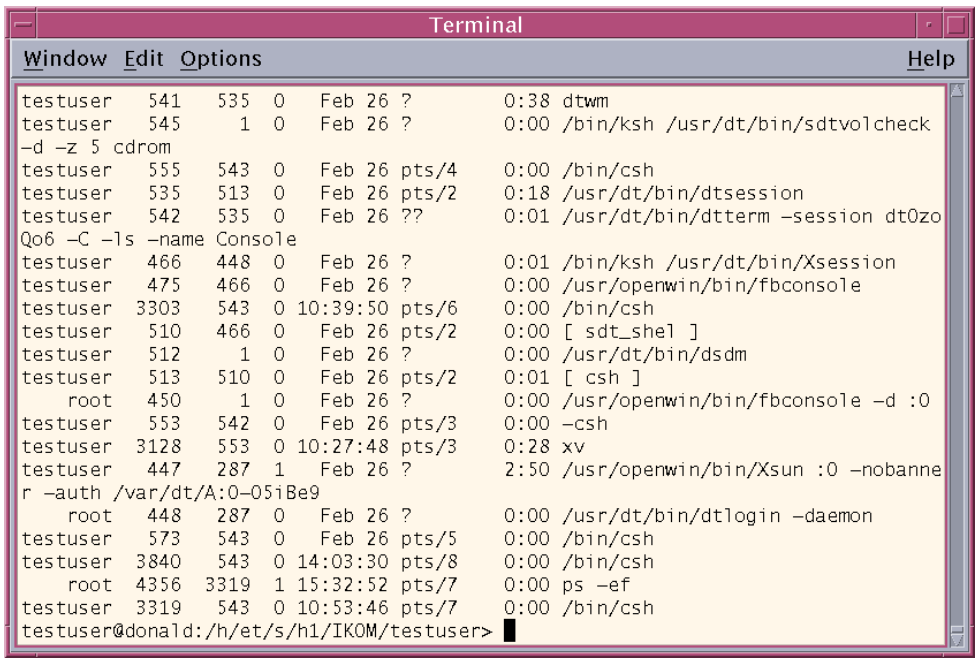

Abbildung 29: Anzeige aller Prozesse

- TTY
  Angaben zum Terminal, mit dem der Prozess kommuniziert.

- TIME
  Die bisher vom Prozess verbrauchte CPU-Zeit.

- CMD
  Das Kommando, mit dem der Prozess gestartet wurde.

Diese Angaben ermöglichen es, die PID für bestimmte Prozesse zu finden (z.B. abgestürzte Programme oder Programme mit hoher CPU-Inanspruchnahme).

9.3.3. Prozess beenden / Signale senden

Mit

```
kill <PID>
```

kann ein Prozess beendet werden.
Hilft dies nicht, kann man es nochmals mit

```
kill -9 <PID>
```

versuchen. <PID> ist hier jeweils durch die PID des Prozesses zu ersetzen, der beendet werden soll.

Das Programm „kill“ sendet ein Signal an einen bestimmten Prozess. Es stehen unterschiedliche Signale zur Verfügung um unterschiedliche Bedingungen zu signalisieren.

Beispielsweise signalisiert das Betriebssystem ein nahendes Herunterfahren, indem allen Prozessen das Signal SIGTERM (15) gesandt wird. Wird beim Aufruf von kill kein Signal angegeben, wird automatisch SIGTERM eingesetzt.
Wird im Terminalfenster CTRL-c gedrückt, während ein Programm läuft, erhält der entsprechende Prozess das Signal SIGINT.
Endet eine Terminalsitzung, erhalten die von diesem Terminal aus gestarteten und mit dem Terminal verbundenen Prozesse das Signal SIGHUP.
Andere Signale informieren beispielsweise einen Prozess darüber, wenn er in eine Pipe schreibt, für die kein lesender Prozess vorhanden ist...

Jeder Prozess kann selbst entscheiden, wie er auf welches Signal reagiert. Allgemein üblich ist jedoch, dass sich jeder Prozess beim Eintreffen des Signales SIGKILL (mit der Signal-Nummer 9) unverzüglich selbst beendet.

10. Übungsaufgaben

Der überwiegende Teil der hier gestellten Übungsaufgaben stammt aus der UNIX-Übung von Dr.-Ing. R. Kamprath.

10.1. Online-Hilfe

1. Schauen Sie sich die Online-Hilfe zum Programm „chdir“ im Terminalfenster an!
2. Schauen Sie sich die Online-Hilfe zum Programm „chmod“ im WWW-Browser an!

10.2. Verzeichnisse

1. Begeben Sie sich in Ihr Homeverzeichnis!
2. Legen Sie ein Unterverzeichnis „uv“ an!
3. Legen Sie im Unterverzeichnis „uv“ weitere Unterverzeichnisse mit den Namen „uv1“ und „uv2“ an!
4. Lassen Sie sich das aktuelle Verzeichnis anzeigen!
5. Welche Bedeutung haben die Zeichen „.“, „..“ und „/“?
6. Wie lautet das Kommando zum Sprung aus dem Homeverzeichnis in das Verzeichnis „/usr/bin“?
7. Wie lautet das Kommando zum Sprung aus der Dateibaum-Wurzel in das Verzeichnis, das dem Homeverzeichnis übergeordnet ist?

10.3. Dateien und Verzeichnisse anzeigen

1. Listen Sie alle Dateien Ihres Homeverzeichnis auf, einschließlich der versteckten Dateien!
2. Listen Sie alle Einträge auf, die mit „a“ beginnen!
3. Listen Sie alle Einträge auf, die aus genau drei beliebigen Zeichen bestehen!
4. Erläutern Sie, welche Zugriffsrechte auf eine Datei bestehen, wenn die Zugriffsrechte als „-rwxr-xr–“ angezeigt werden!
5. Listen Sie den Inhalt des dem Homeverzeichnis übergeordneten Verzeichnisses ohne das Homeverzeichnis zu verlassen!

6. Listen Sie das Verzeichnis eines Nutzers Ihrer Nutzergruppe ohne Ihr Homeverzeichnis zu verlassen!
7. Listen Sie Ihr Homeverzeichnis nachdem Sie zwei Verzeichnisebenen nach oben gewechselt haben!
8. Wieviele Dateien, deren Name mit „f“ beginnt, befinden sich im Verzeichnis „/usr/bin“?
9. Welche ist die größte Datei im Verzeichnis „/usr/bin“, welche die älteste?
10. Listen Sie Ihr Homeverzeichnis samt allen Unterverzeichnissen in Langform!

10.4. Dateien bearbeiten, kopieren und verschieben

1. Erzeugen Sie in Ihrem Homeverzeichnis eine Datei „alf“ mit dem Inhalt „Hund und Katze“.
2. Erzeugen Sie eine Kopie der Datei „alf“ mit dem Namen „ulf“.
3. Leiten Sie das ausführliche Listing des aktuellen Verzeichnisses in eine Datei „inhalt“ und geben Sie sie in der gleichen Kommandozeile aus.
4. Hängen Sie den Inhalt der Datei „ulf“ an die Datei „alf“ an!
5. Erzeugen Sie eine Datei „viermal“ die viermal die Datei „alf“ zum Inhalt hat!
6. Kopieren Sie die Datei „alf“ in das Unterverzeichnis „uv1“!
7. Verschieben Sie die Datei „ulf“ in das Unterverzeichnis „uv1“.
8. Kopieren Sie aus „/usr/bin“ alle Dateien, deren Name mit „h“ beginnt, in das neu anzulegende Unterverzeichnis „kopie“ in „uv1“. Was stellen Sie hinsichtlich Eigentümer, Gruppe, Zugriffsrechten und Modifikationszeitpunkt fest?
9. Worin besteht der Unterschied zwischen den Kommandos „cp“ und „mv“?
10. Versuchen Sie, aus „/usr/bin“ alle Dateien, deren Name mit „h“ beginnt, in Ihr Homeverzeichnis zu verschieben. Warum ist dies nicht möglich?
11. Worin besteht der Unterschied zwischen den Kommandos „ln“ und „cp“?
12. Erzeugen Sie in Ihrem Homeverzeichnis einen Link „elf“, der auf „alf“ verweist.

10.5. Zugriffsrechte

1. Ändern Sie die Zugriffsrechte auf „ulf“ so, dass kein anderer Nutzer irgendwie auf diese Datei zugreifen kann!

10.6. Dateien und Verzeichnisse löschen

1. Löschen Sie die Unterverzeichnisse „uv1“ und „uv2“ mit dem „rmdir“-Befehl! Welches Unterverzeichnis kann gelöscht werden, welches nicht?

2. Löschen Sie die Dateien in den Unterverzeichnissen, so dass Sie die Unterverzeichnisse selbst löschen können!

10.7. Datum und Uhrzeit

1. Geben Sie das aktuelle Datum und die Uhrzeit aus!

2. Ermitteln Sie mit Hilfe des Kommandos „cal“, an was für einem Wochentag Sie geboren wurden!

10.8. Felder aus Zeilen filtern

1. Geben Sie nacheinander jeweils das erste, zweite und dritte Element der Datei „alf“ aus, wobei das Leerzeichen als Trennzeichen verwendet wird!

2. Geben Sie nacheinander jeweils das erste, zweite und dritte Element der Datei „alf“ aus, verwenden Sie „u“ als Delimiter!
   Verwenden Sie anschließend die Option „-f1-2“.

3. Lösen Sie die vorangegangenen zwei Aufgaben unter Verwendung einer Pipe!

4. Geben Sie mit Hilfe der Kommandos „date“ und „cut“ die aktuelle Uhrzeit (und nur diese) aus!

5. Geben Sie nur die Minuten der aktuellen Uhrzeit aus!

6. Versuchen Sie, aus einem mit „cal“ erzeugten Kalender eine Spalte für einen Wochentag herauszufiltern!

10.9. Zeilen in Datei finden

1. In welchen Zeilen (gesucht sind die Zeilennummern) des Online-Manuals zu „date“ kommt die Zeichenfolge „time“ vor?

2. In wievielen Zeilen des Online-Manuals zu „date“ kommt der Begriff „time“ (mit möglicherweise unterschiedlicher Groß- und Kleinschreibung) vor?

3. In wievielen Zeilen komt der Begriff nicht vor?

4. Filtern Sie aus der Kalender-Ausgabe für den August alle die Zeilen heraus, die als zweiten Buchstaben die Zahl „1“ enthalten!

5. Filtern Sie alle diejenigen Zeilen heraus, die mit „1“, „2“ oder „3“ beginnen!
6. Filtern Sie alle diejenigen Zeilen heraus, die nicht mit „1“, „2“ oder „3“ beginnen!
7. Filtern Sie alle diejenigen Zeilen heraus, die entweder „14“ oder „21“ enthalten!
8. Filtern Sie alle anderen Zeilen heraus!
9. Filtern Sie die ersten drei Zeilen aus der Kalenderausgabe für August heraus!
10. Filtern Sie die letzten drei Zeilen heraus!
11. Filtern Sie die fünfte Zeile heraus!

A. Reguläre Ausdrücke

A.1. Überblick

Reguläre Ausdrücke (r.A.) sind Schemata, die beschreiben, wie eine Zeichenfolge aufgebaut ist.
Beim Vergleich einer Zeichenfolge mit einem r.A. kann der r.A. auf die Zeichenfolge zutreffen (match), dies ist der Fall, wenn die Zeichenfolge dem vorgegebenen Schema folgt.
Folgt die Zeichenfolge dem vorgegebenen Schema nicht, ist der r.A. nicht erfüllt.

A.2. Einfache reguläre Ausdrücke

Normale Zeichen - also alle im folgenden Text nicht erwähnten - stehen für sich selbst.
Zeichen, die in regulären Ausdrücken eine Sonderbedeutung haben, muss ein Backslash (\) vorangestellt werden.
Hierzu zählen:

- Punkt („.“)
  Der Punkt steht für ein beliebiges Zeichen außer Newline.
- Backslash (\)
  Der Backslash dient dazu, die Sonderbedeutung des nachfolgenden Zeichens aufzuheben.
- öffnende eckige Klammer („[“)
- Asterisk („*“)
- Circumflex („^“)
  wenn es am Anfang eines regulären Ausdruckes oder direkt nach einer öffnenden eckigen Klammer steht
- Dollar („$“)
- das Zeichen, das den r.A. einschließt.

Eine Folge von Zeichen in eckigen Klammern ([]) repräsentiert ein beliebiges Zeichen dieser Folge, beispielsweise steht „[abcde]“ für eines der Zeichen „a“, „b“, „c“, „d“, „e“.
Ist jedoch das erste Zeichen nach der öffnenden eckigen Klammer ein Circumflex („^“), so wird ein beliebiges *nicht* in der Zeichenfolge enthaltenes Zeichen repräsentiert (außer Newline).
Mit Verwendung des Minus-Zeichens („-“) kann ein Bereich von Zeichen angegeben

werden, bespielsweise „[a-e]“ anstelle von „[abcde]“. Das Minuszeichen verliert seine Sonderbedeutung, wenn es unmittelbar nach der öffnenden bzw. unmittelbar vor der schließenden eckigen Klammer steht.
Eine schließende eckige Klammer beendet die Zeichenfolge nicht, wenn sie unmittelbar auf die öffnende folgt. Der r.A. „[]a-e]“ repräsentiert also eines der Zeichen „]“, „a“, „b“, „c“, „d“, „e“.
In eckigen Klammern haben Punkt, Asterisk, öffnende eckige Klammer und Backslash keine Sonderbedeutung.

Aus den bisher besprochenen Ein-Zeichen-Ausdrücken können reguläre Ausdrücke auf folgende Art und Weise zusammengesetzt werden:

- Folgt auf einen Ein-Zeichen-Ausdruck ein Asterisk („*“), bedeutet das, dass der Ein-Zeichen-Ausdruck 0- oder mehrfach auftritt.

- Folgt auf einen Ein-Zeichen-Ausdruck
  $\{m\}$
  $\{m,\}$
  $\{m,n\}$
  bedeutet dies, dass der Ein-Zeichen-Ausdruck exakt m mal, mindestens m mal oder aber $m \ldots n$ mal auftriftt, wobei m und n nichtnegative Zahlen kleiner als 256 sind.

- Stehen zwei r.A. nacheinander, bedeutet dies, dass im Text nacheinander Zeichenfolgen auftreten, die den jeweils einzelnen r.A. entsprechen.

- Steht ein r.A. in Klammern, muss im Text die Zeichenfolge auftreten, die dem in Klammern stehenden r.A. entspricht.
  Klammern dienen u.a. zur Rückreferenzierung und zum Festlegen der Priorität von Operatoren.

- Ein \n (wobei n eine Dezimalzahl von „1“ bis „9“ ist) trifft genau auf die Textstelle zu, auf die der n-te in Klammern eingeschlossen Ausdruck zutraf, der im selben r.A. *vor* \n stand.
  Dabei wählt n die n-te öffnende Klammer von links aus.

- Ein r.A. kann an Wortgrenzen ausgerichtet werden, „\<“ kennzeichnet einen Wortanfang, „\>“ ein Wortende.

- Ein *gesamter* r.A. kann so festgelegt werden, dass er einen Zeilenanfang, ein Zeilenende bzw. eine gesamte Zeile beschreibt.
  Dabei steht der Circumflex („^“) für den Beginn der Zeile und der Dollar („$“) für das Zeilenende.

A.3. Erweiterte reguläre Ausdrücke

Erweiterte reguläre Ausdrücke bieten zusätzlich folgende Operatoren an:

- Die Zeichen „|“, „+“ und „?“ haben Sonderbedeutung (siehe unten).
- Rückreferenzen werden nicht unterstützt.
- Verankerungen („^“ und „$“) können in Teilausdrücken eingesetzt werden.

Folgt ein Pluszeichen einem Ein-Zeichen-Ausdruck oder einem Ausdruck in Klammern, so bedeutet dies, dass der Text des Ausdruckes ein- oder mehrfach auftritt.

Folgt ein Fragezeichen einem Ein-Zeichen-Ausdruck oder einem Ausdruck in Klammern, so bedeutet dies, dass zum Ausdruck passender Text gar nicht oder einmal auftritt.

Steht ein „|“ zwischen zwei Ausdrücken, bedeutet dies, dass Text auftritt, der zu einem der beiden Ausdrücke passt.