tar, shar, compress and All That

Einführung in die Bereitstellung und Bearbeitung archivierter und komprimierter Software und Dokumentation unter Unix

Wilfried Grieger

Für alle, denen das Fremde nicht gefährlich erscheint und die es deshalb kennenlernen wollen.

Für Daniel, Carsten und Simon, die noch nicht wissen, was Datenverarbeitung ist.

Vorwort

Im Jahr 1931 erschien in New York eine Abhandlung mit dem Titel „1066 and All That“.[1] Die Autoren versuchten darin, grundlegende Gedankengänge zur geschichtsträchtigen Jahreszahl 1066 darzulegen. Seit der Zeit haben andere Verfasser ähnliche Beschreibungen für weitere Themen erarbeitet. In der mathematischen Physik ist zum Beispiel das Buch „PCT, Spin & Statistics, and All That“[2] ein Standardwerk für Studierende der Quantenfeldtheorie geworden.[3]

In dem vorliegenden Buch wird versucht, einige grundlegende Kommandos und Utilities unter Betriebssystemen der Unix-Familie darzustellen, die es erlauben, Files oder sogar ganze Filesysteme in einer übersichtlichen Form zu archivieren. Die zu behandelnden Kommandos, Utilities und ihre engsten Verwandten dienen auch dazu, auf bequeme Weise archivierte Files in ihren Zustand vor der Archivierung zurückzuführen.

Diese Möglichkeiten machen sich alle diejenigen zunutze, die Software oder Dokumentation weltweit über Rechner-

1. W. C. Sellar, R. J. Yeatman, *1066 and All That*, New York 1931
2. R. F. Streater, A. S. Wightman, *PCT, Spin & Statistics, and All That*, New York 1964
3. Weitere neuere Beispiele für „and All That“-Bücher sind:

   P. J. Davis, W. G. Chinn, *3.1416 and all that*, Boston 1985

   J. Hulme, *1789 and all that*, London 1988

   R. Coquereaux, A. Jadczyk, *Riemannian geometry, fiber bundles, Kaluza-Klein theories and all that*, Singapore 1988

netze kopierbar zur Verfügung stellen, ebenso wie diejenigen, die diese Software oder Dokumentation kopieren. Aus diesem Grund muß jeder, der solche Software oder Dokumentation verwenden oder sogar selber bereitstellen möchte, die in dem vorliegenden Buch erläuterten Begriffe kennen.

Die Beschreibungen sind als Einführung gedacht. Experten werden demzufolge detaillierte Erläuterungen spezieller Sachverhalte vermissen, aber diese würden einen Anfänger mehr verwirren als ihm hilfreich sein.

Ich danke allen, die mich bei dem mühevollen Unterfangen, ein Buch zu schreiben, unterstützt haben. Besonders danke ich Helge und Burkhard, deren Kritik immer konstruktiv war. Ebenso danke ich den Herausgebern der Buchreihe „Einführung in die Wissenschaftliche Datenverarbeitung“, Joachim Lammarsch und Helge Steenweg, daß sie das vorliegende Buch als erstes in die Reihe aufgenommen haben.

Das Buch ist auf der Grundlage eines Kurses entstanden, der unter dem gleichen Titel für Benutzer des Rechenzentrums der Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG) zum erstenmal im Juni 1993 gehalten wurde. Es steht unter dem Eindruck des sich zur Zeit vollziehenden Wandels in der wissenschaftlichen Datenverarbeitung.

Sicherlich können deshalb viele Passagen des Buches kritisiert werden, Fehler werden sich eingeschlichen haben. Ich bitte alle diejenigen, die Bemerkungen zu diesem Buch machen möchten, sie auch mir mitzuteilen, damit sie später berücksichtigt werden können.

Göttingen, im August 1993

Wilfried Grieger
wgriege@gwdg.de

Gesellschaft für
wissenschaftliche Datenverarbeitung mbH Göttingen
Am Faßberg
37077 Göttingen

Kurzinhalt

Inhaltsverzeichnis

Tabellenverzeichnis

KAPITEL 1 Grundsätzliches zum Betriebssystem Unix

Das Betriebssystem Unix zählt mittlerweile zu den am weitesten verbreiteten Betriebssystemen auf elektronischen Datenverarbeitungsanlagen. Es wird heute auf PCs, Workstations, Universalrechnern und auf Spezialrechnern, wie zum Beispiel auf Vektor- oder Parallelrechnern, eingesetzt. Es ist damit für nahezu jeden Prozessortyp verfügbar.

Die meisten, die sich mit elektronischer Datenverarbeitung im weiteren Sinn beschäftigen, haben den Begriff „Unix“ zumindest schon einmal gehört. Die wenigsten jedoch werden mit diesem Betriebssystem praktische Erfahrungen gemacht haben, weil der Name „Unix“ nämlich eigentlich nur für ein Betriebssystem, das in der Firma AT&T entwikkelt wurde, verwendet werden darf.

Trotzdem werden in der Regel auch Betriebssysteme anderer Hersteller mit Unix bezeichnet, obwohl sie Neuentwicklungen oder Derivate des ursprünglichen sind. Das führt dazu, daß sich die Derivate in einigen Zügen, die in den meisten Fällen für den Anwender keine große Rolle spielen, unterscheiden.

1.1 Einige Unix-Derivate

In dem vorliegenden Buch wird angestrebt, möglichst nur die Gemeinsamkeiten der Unix-Derivate darzustellen. Dort, wo es angebracht erscheint, werden selbstverständlich Besonderheiten einzelner Derivate erläutert.

Die folgende Aufzählung enthält diejenigen Unix-Derivate, die näher untersucht wurden.

1.1.1 AIX

AIX ist das Unix-Derivat der Firma IBM. Es kann sowohl auf Universalrechnern als auch auf RISC-Workstations, allerdings in unterschiedlichen Varianten, eingesetzt werden.

Es gehört zu den System-V-artigen Derivaten, wurde also vom ursprünglichen Unix von AT&T abgeleitet.

1.1.2 Linux

Linux ist frei verfügbar und kostenlos erhältlich, solange das Copyright der Free Software Foundation nicht verändert wird. Es kann auf Intel-Prozessoren der Baureihen 386 und 486 eingesetzt werden und ist dadurch schon recht häufig anzutreffen. Es enthält alle wichtigen Eigenschaften von Unix.

Es gehört im wesentlichen ebenfalls zu den System-V-artigen Derivaten.

1.1.3 SCO-Unix

SCO-Unix ist von der Firma Santa Cruz Operation entwikkelt worden. Es kann ebenfalls auf 386- und 486-PCs eingesetzt werden.

Es gehört zu den System-V-artigen Derivaten.

1.1.4 SunOS

SunOS ist das Betriebssystem der Firma Sun, das auf deren Workstations eingesetzt wird.

Im Gegensatz zu den obigen Derivaten gehört es zu den BSD-Typen, also einer Abwandlung des ursprünglich von der Berkeley Software Distribution entwickelten Unix-Betriebssystems.

1.1.5 Ultrix

Ultrix wurde von der Firma Digital entwickelt und wird auf deren Workstations eingesetzt.

Es gehört ebenfalls zu den BSD-artigen Derivaten.

1.1.6 Weitere Derivate

Weitere Unix-Derivate, zum Beispiel A/UX von Apple, HP-UX von Hewlett Packard, Xenix von Microsoft, Sinix von Siemens, konnten nicht näher untersucht werden, da die entsprechende Hardware zum Testen nicht zur Verfügung stand.

Sofern kein Zugang zu diesen Unix-Derivaten bestand, ist anhand der Dokumentation versucht worden, die Gemeinsamkeiten mit den obigen Derivaten zu ermitteln.

1.2 Zusammenführung der Derivate

Zur Zeit werden Anstrengungen unternommen, die meisten Unix-Derivate zu vereinheitlichen und gemeinsame Standards zu entwickeln, damit die Vielfalt, die in der Regel den Systemspezialisten die Arbeit erschwert, beendet wird.

Am erfolgversprechendsten scheinen die Bestrebungen der Open Software Foundation (OSF) zu sein, mit dem Betriebssystem OSF/1 einen allgemeinen Unix-Standard einzuführen. Viele Firmen haben bekundet, sich diesem Standard anschließen zu wollen.

Zur Zeit muß OSF/1 wohl noch als Unix-Derivat bezeichnet werden.

1.3 GNU General Public License

In der Regel darf kommerzielle Software nicht frei kopiert oder verändert werden.

Ganz im Gegensatz zu dieser Auffassung steht die von der Free Software Foundation veröffentlichte GNU General Public License ihrer Softwarprodukte. Die GNU General Public License garantiert die freie Verfügbarkeit der Software. Wird die Software weitergegeben, so erhält der Empfänger alle Rechte, die auch der Verteiler besitzt.

Im Text wird die Software, die unter die GNU General Public License fällt, mit dem Zusatz „GNU“ gekennzeichnet.

Auch das Unix-Derivat Linux unterliegt der GNU General Public License.

1.4 Vorausgesetzte Kenntnisse über Unix

Für das Verständnis der in diesem Buch näher behandelten Kommandos und Utilities sind natürlich einige Grundkenntnisse eines Unix-Betriebssystems erforderlich.

Diese Grundkenntnisse kann man sich anhand von Lehrbüchern aneignen. Vorteilhaft ist sicherlich die Möglichkeit des Zugriffs auf einen Rechner, der mit einem Unix-Derivat betrieben wird. Im vorliegenden Buch wird versucht, den Umfang derjenigen Grundkenntnisse, die erforderlich sind, um das Geschriebene zu verstehen, möglichst klein zu halten und bei Bedarf auch Grundlagen zu vermitteln.

Die folgenden Absätze erläutern den Umfang der benötigten Kenntnisse.

1.4.1 Grundlegende Kenntnisse

Das Betriebssystem Unix ist heutzutage in der Regel in der Programmiersprache C geschrieben.

Das Filesystem ist hierarchisch aufgebaut, und zwar mit Directories, Subdirectories und Files. Als Zeichen im Filenamen läßt Unix fast alle 8-Bit-Zeichen zu. Es sollte jedoch bedacht werden, daß es bei der Verwendung von Zeichen im Filenamen, die nicht dem ASCII-Zeichensatz angehören, aber auch bei der Verwendung des Leerzeichens zu unerwünschten und unter verschiedenen Unix-Derivaten auch zu unterschiedlichen Effekten führen kann, wenn ein derartiger Filename einem Kommando oder einem Programm als Parameter übergeben wird.

Die Kommunikation des Anwenders mit dem Betriebssystem geschieht über eine sogenannte Shell, die die eingege-

benen Kommandos und Programmaufrufe interpretiert und die interpretierte Form an das eigentliche Betriebssystem weitergibt. Unter Unix stehen mehrere Shells zur Verfügung. Die bekannteste und älteste ist die nach ihrem Entwickler, Steve Bourne, benannte Bourne-Shell. Die Shell verwaltet die Ausführung der Kommandos oder Programme in Form von Prozessen.

Die Kommunikation mit anderen Rechnern geschieht über ein Rechnernetz, zum Beispiel das Internet.

1.4.2 Grundlegende Kommandos

Zur Verwaltung von Files werden grundlegende Kommandos benötigt. Sie dienen im wesentlichen zum Anlegen und Löschen von Files und Filesystemen. Ebenso sind zum Ausführen von Programmen und zur Kommunikation mit anderen weitere Kommandos erforderlich.

Als Beispiele seien die folgenden einfachen Kommandos aufgezählt:

- zum Löschen von Files: **`rm`**
- zum Umbenennen oder Verschieben von Files: **`mv`**
- zum Kopieren von Files: **`cp`**
- zum Anzeigen von Fileinhalten: **`cat`** oder **`more`**
- zum Einrichten einer Subdirectory: **`mkdir`**
- zum Löschen einer Subdirectory: **`rmdir`**
- zum Anzeigen einer Liste von Files in einer Subdirectory: **`ls`**
- zum Wechseln der Working-Directory: **`cd`**
- zur Installation von Programmen: **`make`**
- zur Eröffnung einer neuen Bourne-Shell: **`sh`**
- zum Kopieren von Files von anderen Rechnern: **`ftp`**
- zur Kommunikation mit anderen: **`mail`**

In der Regel können jedem Kommando oder Programm Optionen oder Parameter angefügt werden, die den Ablauf des Prozesses beeinflussen. Kommandos und Programmaufrufe können zu Pipes zusammengefaßt werden.

Kommandos und Programmaufrufe können in ein Shellscript geschrieben werden. Der Aufruf des Shellscripts wird von der Shell als neues Kommando interpretiert und vom Betriebssystem ausgeführt.

Die von Kommandos oder Programmen erwarteten Eingaben müssen nicht vom Bildschirm aus erfolgen, sondern können umgelenkt werden. Ebenso müssen die Ausgaben von Kommandos oder von Programmen nicht auf dem Bildschirm erscheinen, sondern können ebenfalls umgelenkt werden.

1.4.3 Texteditoren

Zur Bearbeitung von Files stehen unter Unix einige Texteditoren zur Verfügung. Standardmäßig sind im Betriebssystem zumindest

- der Zeileneditor **`ex`** und
- der Full-Screen-Editor **`vi`**

vorhanden.

Eine größere Funktionsvielfalt bietet der von der Free Software Foundation entwickelte

- GNU **`emacs`**,

der jedoch nicht auf allen Rechnern installiert ist.

Die grundlegende Beherrschung eines Texteditors ist nicht zwingend erforderlich, erleichtert aber die Manipulation von Files erheblich.

1.4.4 Usenet-Newssystem

Das Usenet-Newssystem ist ein internationaler Kommunikationsdienst, bei dem Diskussionsbeiträge, Software oder Dokumentation in Form von sogenannten Artikeln innerhalb

thematisch geordneter Newsgroups zwischen vielen Rechnern auf der ganzen Welt ausgetauscht werden.

Zum Lesen, Empfangen oder Senden einzelner Artikel und zum Antworten werden Newsreader verwendet, die mittlerweile für viele unterschiedliche Betriebs-systeme zur Verfügung stehen.

Häufig wird das Usenet-Newssystem auch mit NetNews oder einfach nur mit News bezeichnet.

1.5 Zur verwendeten Notation

Als Filesystem wird eine Menge von Subdirectories mit darunterliegenden weiteren Subdirectories oder Files in einer beliebigen Zahl von hierarchischen Stufen bezeichnet.

Einzugebende Kommandos werden in der folgenden Form geschrieben:

`rm -i` *`filename`*

Dabei ist **`rm`** das Unix-Kommando, **`-i`** eine der möglichen angebbaren Optionen zum Kommando **`rm`** und *`filename`* durch einen geeigneten Parameter, in diesem Fall also durch den Namen eines Files, zu ersetzen. Diese Form wird dann gewählt, wenn der Parameter nicht näher bestimmt werden soll.

Soll das File mit dem Namen „testfile“ interaktiv gelöscht werden, so wird das zugehörige Kommando in der Form angegeben, wie es auf der Tastatur eingegeben werden müßte:

```
rm -i testfile
```

Eine Bildschirmausgabe wird in der folgenden Form dargestellt:

Die Kommandofolge

```
cd $HOME
ls -l
```

liefert am Bildschirm folgende Ausgabe:

```
total 16
drwx------  3 wgriege       512 Apr 13 15:18 Beispiel
drwx------  2 wgriege       512 Apr 10 08:37 Noch_eine_Directory
drwx------  2 wgriege       512 Apr 10 08:36 Weitere_Directory
-rw-------  1 wgriege      1675 Apr 10 08:43 ein_file
```

Die Rückmeldungen des Betriebssystems, die System-Prompts, werden hier nicht dargestellt, da ihr Aussehen vom zugrundeliegenden Betriebssystem und der verwendeten Shell abhängt.

KAPITEL 2 Zur Auswahl der behandelten Kommandos und Utilities

Die zu behandelnden Kommandos sollen dazu dienen, Software oder Dokumentation in einer komfortablen Weise anderen unter Unix zur Verfügung zu stellen oder kopierte Software oder Dokumentation wieder so aufzubereiten, daß sie eingesetzt bzw. gedruckt werden kann.

Dafür gibt es zwei grundsätzlich verschiedene Verfahren. Zum einen kann die Software oder Dokumentation auf einem Rechner bereitliegen, so daß sie zum Beispiel mit dem Unix-Kommando **`ftp`** über das Internet kopiert werden kann. Zum anderen kann sie über ein Kommunikationssystem mit **`mail`** oder über das Usenet-Newssystem verschickt werden.

Aus diesem Grund müssen in der Regel auch zwei grundsätzlich verschiedene Kommandofamilien benutzt werden. Diese werden in den beiden folgenden Abschnitten näher erläutert. Der dritte Abschnitt enthält weitere nützliche Kommandos, die noch zusätzlich verwendbar sind und Vorteile in verschiedenen Bereichen bringen.

2.1 `tar` und Verwandte

In der Regel besteht Software aus einem Filesystem mit einem Makefile, in dem die zur Installation der Software benötigten Eigenschaften definiert sind. Mit Hilfe des Kommandos **`make`** wird dann in der Regel die Software installiert.

Wenn andere die Software kopieren sollen, so ist es sicherlich zeitaufwendig, jedes einzelne File in dem Filesystem zu kopieren. Einfacher ist es, wenn das gesamte Filesystem in

einem einzigen File, dem sogenannten Archiv-File, zusammengefaßt ist.

Diese Zusammenfassung kann durch das Kommando **tar** bewerkstelligt werden.

Andere Kommandos, die dem gleichen Zweck dienen, aber andere oder weitergehende Funktionalitäten besitzen, sind **ar**, **mdtar** und das von der Free Software Foundation entwickelte GNU **gtar**.

Es lassen sich sicherlich in diesem Zusammenhang noch weitere Utilities aufzählen, doch werden sie in diesem Buch nicht berücksichtigt, damit die Übersichtlichkeit erhalten bleibt.

2.2 **shar** und Verwandte

Falls das direkte Kopieren von Software oder Dokumentation von einem Rechner zu einem anderen über Rechnernetze nicht möglich ist, kann die Software oder Dokumentation auch über die elektronische Kommunikation, die Electronic Mail, abgekürzt E-Mail, oder das Usenet-Newssystem verschickt werden. Auch in diesem Fall ist es sinnvoll, ein Filesystem in einem einzigen File zusammenzufassen, um nicht jedes File einzeln senden zu müssen und trotzdem die Information über die Struktur des Filesystems zu erhalten.

Da einige E-Mail-Systeme Besonderheiten aufweisen, die es nicht erlauben, den vollständigen 8-Bit-Zeichensatz, der in der Regel von dem Kommando **tar** ausgenutzt wird, zu übertragen, wurde für das Zusammenfassen von Filesystemen eine andere Utility entwickelt, nämlich das Programm **shar**. Es handelt sich hierbei nicht um ein im Unix-Betriebssystem verankertes Kommando, sondern ein in der Regel in der Programmiersprache C geschriebenes Programm, das im folgenden als Utility bezeichnet wird.

shar wird von mehreren Entwicklern angeboten, es handelt sich um ein Public-Domain-Produkt. Demzufolge kann es

auch in einigen Eigenschaften von den hier beschriebenen abweichen.

shar erzeugt aus einem Filesystem ein Shellscript, das zusammen mit dem Kommando **sh** wieder das ursprüngliche Filesystem erzeugen kann.

2.3 compress und Verwandte

Um beim Kopiervorgang über stark beanspruchte Rechnernetze möglichst wenig Zeichen übertragen zu müssen, können Files vor dem Kopieren komprimiert werden.

Für das Komprimieren von Files stehen verschiedene Verfahren zur Verfügung. Eines davon wird vom Betriebssystemkommando **compress** verwendet. Als Alternative kann die Utility GNU **gzip** von der Free Software Foundation benutzt werden, die sich ein anderes Verfahren zunutze macht.

Zum Dekomprimieren von Files können ebenfalls die Kommandos **compress** bzw. die Utility GNU **gzip** verwendet werden.

Als Kompressionsprogramme stehen noch eine ganze Reihe weiterer Utilities im Public-Domain-Bereich zur Verfügung, die für Spezialanwendungen sicherlich Vorteile gegenüber den oben erwähnten haben, aber hier nicht näher untersucht werden sollen, da sie noch nicht so weit verbreitet sind wie die anderen.

Soll ein File, das den vollständigen 8-Bit-Zeichensatz ausnutzt, also zum Beispiel ein Binär-File, über E-Mail oder das Usenet-Newssystem verschickt werden, so ist dies in der Regel nicht immer fehlerfrei möglich. Es gibt jedoch Kommandos, die den verwendeten Zeichensatz im File so umwandeln können, daß das damit modifizierte File über E-Mail oder das Usenet-News-system unbeschadet versendet und beim Empfänger in den Originalzustand zurückversetzt werden kann.

Es handelt sich hierbei um die Kommandos **uuencode** und **uudecode**, die in Unix-Betriebssystemen standardmäßig vorhanden sind. Obwohl diese also keine Komprimierungsprogramme sind, sollen sie aber trotzdem an dieser Stelle mit aufgeführt werden. Mit Hilfe dieser Programme können nämlich auch Filesysteme, die mit **tar** archiviert wurden, über E-Mail oder das Usenet-Newssystem verschickt werden. Somit kommt es bei der Entscheidung, welches Archivierungsprogramm verwendet werden soll, ob **tar** oder **shar**, nicht mehr in erster Linie darauf an, ob das Archiv später kopiert oder über E-Mail oder das Usenet-Newssystem versendet wird.

KAPITEL 3 Anwendungsbeispiel

Die behandelten Kommandos sollen in den folgenden Kapiteln anhand eines Anwendungsbeispiels vorgestellt werden. Zur Darstellung der Bildschirmausgaben sei erwähnt, daß diese von einer DECstation 5000 unter dem Betriebssystem Ultrix erzeugt wurden.

Das Anwendungsbeispiel besteht aus einem File-system, das nun näher erläutert wird.

3.1 Das Filesystem

Bei dem im Anwendungsbeispiel betrachteten File-system handelt es sich um die Directory „Beispiel“, die in die Home-Directory der Userid „wgriege“ eingetragen ist.

Die Login-Shell dieser Userid ist die Korn-Shell, unter der die standardmäßigen Zugriffsrechte auf Files oder Directories durch das Unix-Kommando

```
umask 077
```

geregelt sind. Wird die Home-Directory mit dem Kommando

```
cd $HOME
```

zur Working-Directory gemacht, so liefert das danach eingegebene Kommando

```
ls -l
```

am Bildschirm folgende Ausgabe:

```
total 16
drwx------  3 wgriege       512 Apr 13 15:18 Beispiel
drwx------  2 wgriege       512 Apr 10 08:37 Noch_eine_Directory
drwx------  2 wgriege       512 Apr 10 08:36 Weitere_Directory
-rw-------  1 wgriege      1675 Apr 10 08:43 ein_file
```

Wird „Beispiel“ mit dem Kommando

```
cd Beispiel
```

zur Working-Directory gemacht, so befindet sich dort eine Subdirectory „Unterdirectory“ und die beiden Files „erstes_file“ und „zweites_file“, die nach der Eingabe des Kommandos

```
ls -l
```

am Bildschirm angezeigt werden:

```
total 12
drwx------  2 wgriege       512 Apr 13 15:17 Unterdirectory
-rw-------  1 wgriege       199 Apr 13 15:16 erstes_file
-rw-------  1 wgriege       201 Apr 10 09:43 zweites_file
```

In die Subdirectory „Unterdirectory“ ist nur noch ein weiteres File, nämlich „drittes_file“ eingetragen, so daß die Kommandos

```
cd Unterdirectory
ls -l
```

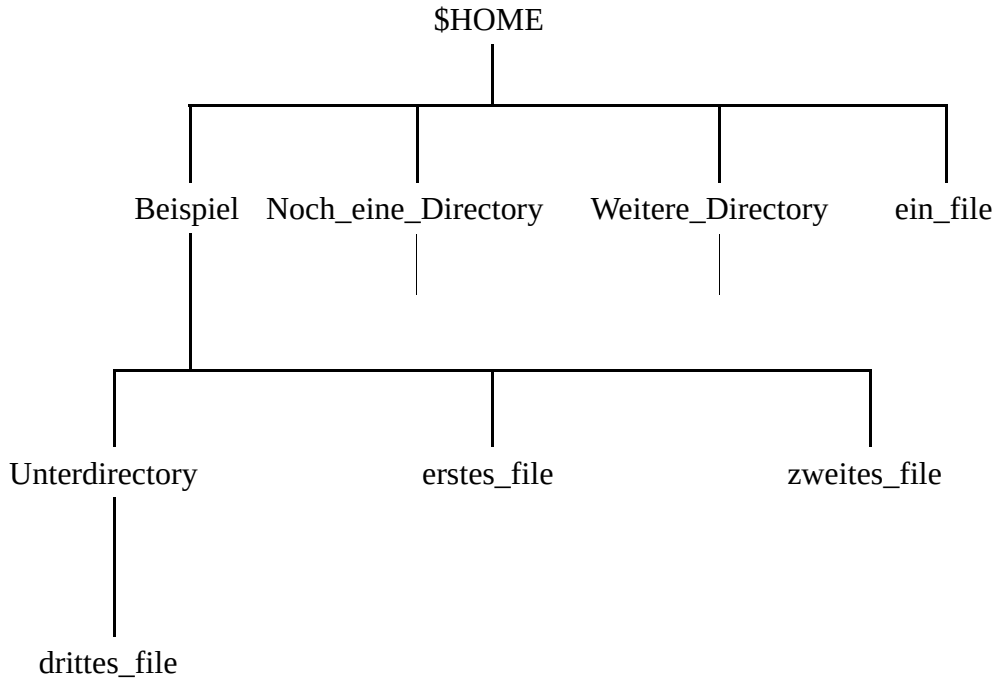
folgende Ausgabe am Bildschirm bewirken:

```
total 4
-rw-------  1 wgriege       198 Apr 13 15:17 drittes_file
```

Dieses Beispiel ist recht einfach, aber es ist bewußt so gewählt, damit die Übersichtlichkeit nicht verlorengeht.

3.2 Struktur des Filesystems

Anhand des folgenden Bildes soll noch einmal die Struktur des Filesystems anschaulich zusammengefaßt werden.



3.3 Inhalt der Files

Für die Behandlung der Kommandos ist eigentlich die Kenntnis der Inhalte der vier Files „ein_file“, „erstes_file“, „zweites_file“ und „drittes_file“ nicht erforderlich.

Da jedoch auch auf die Inhalte der Archiv-Files, Shellscript-Archive und der komprimierten Files eingegangen werden soll, ist es sicherlich hilfreich, vorher die Inhalte der einzelnen Files zu kennen.

Das Kommando zum Anzeigen des Inhalts des Files „ein_file“

```
cat ein_file
```

liefert die Bildschirmausgabe:

```
# In diesem File stehen einige v"ollig belanglose Daten. Es dient
# lediglich zum F"ullen der Beschreibung.

Anhand dieses Beispiels soll sp"ater die Komprimierung von Files
demonstriert werden. Deshalb braucht der folgende Text nicht
gelesen zu werden.

abc defg hijkl mnopqr stuvwxy zab cdef ghijk lmnopr stuvwx yz 12 34
567 890 eins zwei drei vier f"unf sechs sieben acht neun zehn elf
zw"olf dreizehn

abc defg hijkl mnopqr stuvwxy zab cdef ghijk lmnopr stuvwx yz 12 34
567 890 eins zwei drei vier f"unf sechs sieben acht neun zehn elf
zw"olf dreizehn

abc defg hijkl mnopqr stuvwxy zab cdef ghijk lmnopr stuvwx yz 12 34
567 890 eins zwei drei vier f"unf sechs sieben acht neun zehn elf
zw"olf dreizehn

abc defg hijkl mnopqr stuvwxy zab cdef ghijk lmnopr stuvwx yz 12 34
567 890 eins zwei drei vier f"unf sechs sieben acht neun zehn elf
zw"olf dreizehn

abc defg hijkl mnopqr stuvwxy zab cdef ghijk lmnopr stuvwx yz 12 34
567 890 eins zwei drei vier f"unf sechs sieben acht neun zehn elf
zw"olf dreizehn

abc defg hijkl mnopqr stuvwxy zab cdef ghijk lmnopr stuvwx yz 12 34
567 890 eins zwei drei vier f"unf sechs sieben acht neun zehn elf
zw"olf dreizehn

abc defg hijkl mnopqr stuvwxy zab cdef ghijk lmnopr stuvwx yz 12 34
567 890 eins zwei drei vier f"unf sechs sieben acht neun zehn elf
zw"olf dreizehn

abc defg hijkl mnopqr stuvwxy zab cdef ghijk lmnopr stuvwx yz 12 34
567 890 eins zwei drei vier f"unf sechs sieben acht neun zehn elf
zw"olf dreizehn

abc defg hijkl mnopqr stuvwxy zab cdef ghijk lmnopr stuvwx yz 12 34
567 890 eins zwei drei vier f"unf sechs sieben acht neun zehn elf
zw"olf dreizehn

01234567890123

# Das ist nun das Ende des Files "ein_file".
```

Das Kommando zum Anzeigen des Inhalts des Files „erstes_file“

`cat erstes_file`

liefert die Bildschirmausgabe:

```
# Dieses ist der Inhalt des ersten Files in der Directory
# "Beispiel".
# Der Text ist f"ur das folgende irrelevant.
# Er dient nur zur Illustration.

# Damit endet das File "erstes_file".
```

Das Kommando zum Anzeigen des Inhalts des Files „zweites_file“

```
cat zweites_file
```

liefert die Bildschirmausgabe:

```
# Dieses ist der Inhalt des zweiten Files in der Directory
# "Beispiel".
# Der Text ist f"ur das folgende irrelevant.
# Er dient nur zur Illustration.

# Damit endet das File "zweites_file".
```

Das Kommando zum Anzeigen des Inhalts des Files „drittes_file“

```
cat drittes_file
```

liefert die Bildschirmausgabe:

```
# Dieses ist der Inhalt des ersten Files in der Subirectory
# "Unterdirectory".
# Der Text ist f"ur das folgende irrelevant.
# Er dient nur zur Illustration.

# Damit endet das File "drittes_file".
```

Bei Bedarf werden zur Behandlung von Einzelaspekten noch weitere Files in das Filesystem des Anwendungsbeispiels hineinkopiert. Deren Inhalte werden dann jeweils gesondert betrachtet.

KAPITEL 4 `tar` - zur Archivierung von Files und Filesystemen

Die Archivierung von Files oder Filesystemen spielt heute keine so große Rolle mehr wie früher, als der Magnetplattenplatz einer elektronischen Datenverarbeitungsanlage nur in einem beschränkten Maß den Anwendern zur freien Benutzung zur Verfügung stand.

Heute ist der Magnetplattenplatz im Vergleich zu anderen Rechnerbetriebsmitteln eine billige Ressource, die damit nahezu unbeschränkt einsetzbar ist.

Trotzdem soll im folgenden zunächst einmal der Zweck einer Archivierung von Files oder Filesystemen erläutert werden und danach der Wandel ihrer Bedeutung.

4.1 Zweck einer Archivierung

Ein Filesystem, das unter dem Betriebssystem Unix angelegt ist, belegt auf den am Rechnersystem angeschlossenen Magnetplatten einen Platz, der nicht mehr zur Speicherung anderer Files verwendet werden kann.

Wird das Filesystem nicht mehr so häufig benötigt wie andere Files, so kann es auf andere Datenträger, zum Beispiel auf ein Magnetband, kopiert werden. Das geschieht am einfachsten dadurch, daß jedes File einzeln kopiert wird. Anschließend kann das Filesystem auf der Magnetplatte gelöscht werden, so daß der freigewordene Platz von anderen dringend benötigten Files belegt werden kann. Der gesamte Vorgang wird als *Archivierung* bezeichnet.

Das Kopieren der einzelnen Files auf einen anderen Datenträger hat den Nachteil, daß das ursprüngliche Filesystem in

seiner Struktur nur dann wiederhergestellt werden kann, wenn die Struktur ebenfalls bekannt ist.

Das Unix-Kommando **`tar`**, das unter manchen Unix-Derivaten auch als *tape archiver* bezeichnet wird, übernimmt beide Aufgaben. Es sorgt für die Archivierung der einzelnen Files und vermerkt gleichzeitig die Struktur des Filesystems auf dem neuen Datenträger, so daß es bei Bedarf ohne großen Aufwand wiederhergestellt werden kann.

4.2 Wandel in der Bedeutung der Archivierung

Der Magnetplattenplatz ist heute so billig geworden, daß eine Archivierung von Filesystemen aus Kostengründen im Vergleich zu früher kaum noch erforderlich ist. Die Archivierung kann jedoch benutzt werden, um komplexe Filesysteme in einer übersichtlichen Form zu speichern und anderen eventuell zum Kopieren bereitzustellen.

Das Unix-Kommando **`tar`** erzeugt nämlich aus einem ganzen Filesystem ein einziges Archiv-File, so daß das lästige Kopieren vieler Files durch das einfache Kopieren nur eines Files ersetzt wird, aber trotzdem die Information über die Struktur des gesamten File-systems erhalten bleibt. Software, die zum Kopieren auf anderen Rechnern bereitliegt, wird deshalb heute in der Regel in der mit **`tar`** archivierten Form geliefert. Auf dem eigenen Rechner kann sie dann ebenfalls mit **`tar`** wieder in die Ursprungsform zurückverwandelt werden.

Die Archivierung von Files oder Filesystemen auf andere Datenträgern als auf Magnetplatten wird hier nicht weiter behandelt, da sie in der Regel nur von Systemadministratoren durchgeführt wird oder durchgeführt werden kann. Die Beschreibung dieser Art der Archivierung würde den Rahmen der Einführung sprengen.

4.3 Die Funktionen des Kommandos `tar`

In den folgenden Abschnitten werden die fünf Funktionen des Unix-Kommandos **tar** beschrieben. Diese werden durch fünf Grundoptionen, in manchen Unix-Derivaten auch als *function keys*, *function letters* oder *required flags* bezeichnet, gewählt, von denen genau eine, und zwar als *erste Option* anzugeben ist. Diese fünf Grundoptionen sind:

- **-c** zum Erzeugen („create“) eines neuen Archivs,
- **-r** zum Erweitern („write“) eines Archivs mit neuen Files,
- **-t** zum Anzeigen („type“) aller Files in einem Archiv,
- **-u** zum Austauschen („update“) von Files in einem Archiv und
- **-x** zum Extrahieren („extract“) von Files aus einem Archiv.

Wie es unter dem Betriebssystem Unix üblich ist, können weitere Optionen mit diesen Grundoptionen zu einer Zeichenkette **-***kette* zusammengefaßt werden.

4.3.1 Erzeugung eines Archivs

Um ein neues Archiv mit dem Kommando **tar** anzulegen, muß die Grundoption **-c** gewählt werden.

Anlegen eines Archivs

Anhand des Beispiels im Kapitel „Anwendungsbeispiele“ soll nun das Anlegen eines neuen Archivs, also die Erzeugung eines Archiv-Files, vielfach auch tar-File genannt, erläutert werden.

Die Kommandofolge

```
cd $HOME
ls -l
```

liefert am Bildschirm die Ausgabe:

```
total 16
drwx------  3 wgriege       512 Apr 13 15:18 Beispiel
drwx------  2 wgriege       512 Apr 10 08:37 Noch_eine_Directory
drwx------  2 wgriege       512 Apr 10 08:36 Weitere_Directory
-rw-------  1 wgriege      1675 Apr 10 08:43 ein_file
```

Das Filesystem „Beispiel“ soll in das zu erstellende Archiv-File „Beispiel.tar“ kopiert werden. Dazu wird

```
tar -cvf Beispiel.tar Beispiel
```

eingegeben. Die Option **-v** steht für „verbose“ und bedeutet, daß das Kommando **tar** seine Aktivitäten ausführlich am Bildschirm meldet. Ohne diese Option arbeitet das Kommando ohne Bildschirmausgabe. Die Option **-f** steht dabei für „file“, so daß die Archivierung in das in der Kommandozeile unmittelbar dahinter angegebene File „Beispiel.tar“ erfolgt, im Gegensatz zu einer Archivierung auf ein Magnetband.

Wird das Kommando in der obigen Form eingegeben, erscheinen danach am Bildschirm die Meldungen:

```
a Beispiel/
a Beispiel/Unterdirectory/
a Beispiel/Unterdirectory/drittes_file 1 blocks
a Beispiel/erstes_file 1 blocks
a Beispiel/zweites_file 1 blocks
```

Der Buchstabe „a“ zu Beginn jeder Zeile deutet an, daß es sich um einen Archivierungsvorgang handelt. Alle Files des betrachteten Filesystems „Beispiel“ sind demnach archiviert worden.

Das Kommando

```
ls -l
```

zur Auflistung der Files in der Working-Directory zeigt danach auch das Archiv-File „Beispiel.tar“ an:

```
total 28
drwx------  3 wgriege       512 Apr 13 15:18 Beispiel
-rw-------  1 wgriege     10240 Apr 15 08:37 Beispiel.tar
drwx------  2 wgriege       512 Apr 10 08:37 Noch_eine_Directory
drwx------  2 wgriege       512 Apr 10 08:36 Weitere_Directory
-rw-------  1 wgriege      1675 Apr 10 08:43 ein_file
```

Zur Nomenklatur

Das Kommando **tar** läßt als Namen des Archiv-Files jeden unter Unix gültigen Filenamen zu. Trotzdem hat es sich eingebürgert, für den Namen des Archiv-Files denjenigen des zu archivierenden Filesystems zu wählen und an ihn die Endung **.tar** anzuhängen. Dieser Brauch soll auch im folgenden beibehalten werden.

Damit sieht die allgemeine Syntax des Kommandos zum Anlegen eines Archivs folgendermaßen aus:

tar -cvf *filename*.**tar** *filename*

filename ist dann durch den Namen der Directory des zu archivierenden Filesystems zu ersetzen. Er kann als absoluter oder relativer Pfadname angegeben werden. Im Archiv wird er jedoch ausschließlich als relativer Pfadname registriert, das heißt, beginnt der Pfadname mit einem „/“, so wird dieses Zeichen im Archiv weggelassen.

Die Reihenfolge der anzugebenden Filenamen, nämlich zuerst das File, in das archiviert, dann erst das Filesystem, das archiviert werden soll, erscheint zunächst ungewöhnlich oder sogar falsch. Sie wird jedoch verständlich, wenn man berücksichtigt, daß unmittelbar hinter der Option **-f** der Name des Archiv-Files folgen muß.

Stellt man die Optionen und Parameter etwas um, zum Beispiel folgendermaßen:

tar -cv *filename* **-f** *filename*.**tar**

so erscheint die Reihenfolge der anzugebenden File-namen natürlicher. Das Kommando **tar** unterstützt beide Notationen.

Im folgenden soll jedoch auch weiterhin die erste Form gewählt werden, da sie die am weitesten verbreitete ist.

Inhalt eines Archiv-Files

Für das Verständnis des Geschriebenen ist es nicht erforderlich, den Aufbau und den Inhalt eines Archiv-Files zu kennen. Auf keinen Fall sollte dieser Inhalt mit Hilfe eines Texteditors verändert werden.

Um trotzdem einmal den Inhalt eines Archiv-Files zu zeigen, wird er anhand des Files „Beispiel.tar“ vorgestellt. Viele von **`tar`** eingefügte Zeichen sind vom Bildschirm nicht darstellbar, deshalb sind sie in der folgenden Bildschirmausgabe durch *`[...]`* ersetzt. Das Kommando

```
cat Beispiel.tar
```

zur Anzeige des Inhalts des Files „Beispiel.tar“ liefert am Bildschirm nach dieser Ersetzung:

```
Beispiel/[...]Beispiel/Unterdirectory/[...]Beispiel/Unterdirectory/
drittes_file[...]# Dieses ist der Inhalt des ersten Files in der
Subirectory
# "Unterdirectory".
# Der Text ist f"ur das folgende irrelevant.
# Er dient nur zur Illustration.

# Damit endet das File "drittes_file".
[...]Beispiel/erstes_file[...]# Dieses ist der Inhalt des ersten
Files in der Directory
# "Beispiel".
# Der Text ist f"ur das folgende irrelevant.
# Er dient nur zur Illustration.

# Damit endet das File "erstes_file".
[...]Beispiel/zweites_file[...]# Dieses ist der Inhalt des zweiten
Files in der Directory
# "Beispiel".
# Der Text ist f"ur das folgende irrelevant.
# Er dient nur zur Illustration.

# Damit endet das File "zweites_file".
[...]
```

Der Inhalt der einzelnen Files ist demnach dort aufgeführt und zusätzlich noch die Struktur des archivierten Filesystems in einer verschlüsselten Form. Der Inhalt eines Directory-Files ist im Archiv-File *nicht* verzeichnet.

4.3.2 Erweitern eines Archivs

Das Archiv-File „Beispiel.tar“ soll nun noch um das in der Home-Directory befindliche File „ein_file“ ergänzt werden. Als Grundoption muß dazu **-r** gewählt werden. Demnach kann das Kommando zum Erweitern eines bestehenden Archivs folgendermaßen aussehen:

```
tar -rvf Beispiel.tar ein_file
```

Die Option **-v** bewirkt ausführliche Rückmeldungen des Kommandos **tar**. Die Option **-f** bedeutet wieder, daß die Archivierung in das File „Beispiel.tar“, dessen Name als Parameter unmittelbar hinter der Option angegeben wird, und nicht auf ein Magnetband erfolgt.

Am Bildschirm erscheint nach der Eingabe des Kommandos in der obigen Form:

```
a ein_file 4 blocks
```

Der Buchstabe „a“ in der ersten Spalte deutet wieder an, daß es sich um eine Form der Archivierung handelt. Damit ist das File „ein_file“ im Archiv-File „Beispiel.tar“ aufgenommen worden.

Mit einem Kommando können auch mehrere Files oder Filesysteme *file1*, *file2*, *... filen* in ein bestehendes Archiv integriert werden. Die allgemeine Syntax sieht dann folgendermaßen aus:

tar -rvf *filename*.**tar** *file1 file2* **...** *... filen*

Als Filenamen sind wieder absolute oder relative Pfadnamen zulässig. Beginnt jedoch ein Filename mit einem „/“, so wird dieses Zeichen im Archiv weggelassen.

4.3.3 Inhaltsverzeichnis eines Archivs

Damit **tar** das Inhaltsverzeichnis eines Archiv-Files ausgibt, muß die Grundoption **-t** gewählt werden. Für die Ausgabe eines Inhaltsverzeichnisses des Archiv-Files „Beispiel.tar“ auf dem Bildschirm kann demnach folgendes eingegeben werden:

```
tar -tf Beispiel.tar
```

Die Option **-f** bewirkt wieder, daß das Inhaltsverzeichnis des Archiv-Files „Beispiel.tar“, dessen Name als Parameter unmittelbar hinter der Option angegeben wird, ausgegeben wird und nicht dasjenige eines Archivs auf einem Magnetband. Die Option **-v** ist hier nicht erforderlich, da **-t** schon eine Bildschirmausgabe bewirkt.

Wird das Kommando in der obigen Form eingegeben, so erscheinen am Bildschirm die Meldungen:

```
Beispiel/
Beispiel/Unterdirectory/
Beispiel/Unterdirectory/drittes_file
Beispiel/erstes_file
Beispiel/zweites_file
ein_file
```

Die allgemeine Syntax des Kommandos **tar** zum Anzeigen des Inhaltsverzeichnisses eines Archiv-Files ist demzufolge:

tar -tf *filename*.**tar**

wobei *filename*.**tar** mit absolutem oder relativem Pfadnamen angegeben werden kann.

4.3.4 Austausch von Files im Archiv

Nach der Archivierung des Files „drittes_file“ aus der Subdirectory „Unterdirectory“ ist es verändert worden. Die Kommandofolge

```
cd $HOME
cat Beispiel/Unterdirectory/
                              drittes_file
```

zeigt am Bildschirm:

```
# Dieses ist der Inhalt des ersten Files in der Subdirectory
# "Unterdirectory".
# Der Text ist f"ur das folgende irrelevant.
# Er dient nur zur Illustration.

# Er ist nun ver"andert worden.

# Damit endet das File "drittes_file".
```

Das File „drittes_file“ im Archiv-File „Beispiel.tar“ soll nun durch dieses neue ersetzt werden. Dazu muß die Grundoption **-u** gewählt werden. Damit kann das Kommando zum Austausch folgendermaßen aussehen:

```
tar -uvf Beispiel.tar
  Beispiel/Unterdirectory/drittes_file
```

Die Option **-f** dient wieder zum Austausch im Archiv-File „Beispiel.tar“, dessen Name als Parameter wieder unmittelbar hinter dieser Option angegebenen wird, und nicht auf einem Magnetband. Die Option **-v** bewirkt die folgende Bildschirmausgabe:

```
a Beispiel/
a Beispiel/Unterdirectory/
a Beispiel/Unterdirectory/drittes_file 1 blocks
```

Der Buchstabe „a“ deutet wieder an, daß es sich um einen Archivierungsvorgang handelt. Zusätzlich zum Inhalt des Files wird noch vermerkt, in welcher Subdirectory es sich vor der Archivierung befunden hat. Das neue File wird nur dann ins Archiv aufgenommen, wenn das Änderungsdatum neuer als das im Archiv-File ist. Wenn das zu ersetzende File nicht mit dem Filenamen im Archiv übereinstimmt, also im Archiv nicht unter dem angegebenen Namen vorhanden ist, so wirkt die Grundoption **-u** wie die Grundoption **-r**, die

bewirkt, daß das Archiv durch das eigentlich zu ersetzende File erweitert wird.

In einer Kommandozeile können auch mehrere Files oder ganze Filesysteme im Archiv ersetzt werden. Die allgemeine Kommandosyntax lautet dafür:

tar -uvf *filename*.**tar** *file1 file2* **...**
... filen

Das Inhaltsverzeichnis des Archiv-Files „Beispiel.tar“ mit dem Kommando

tar -tf Beispiel.tar

sieht nun folgendermaßen aus:

```
Beispiel/
Beispiel/Unterdirectory/
Beispiel/Unterdirectory/drittes_file
Beispiel/erstes_file
Beispiel/zweites_file
ein_file
Beispiel/
Beispiel/Unterdirectory/
Beispiel/Unterdirectory/drittes_file
```

Die beiden im Archiv-File verzeichneten Files „Beispiel/Unterdirectory/drittes_file“ lassen sich anhand dieser Ausgabe nicht unterscheiden. Erst die Ausgabe des Kommandos

tar -tvf Beispiel.tar

zeigt, daß das zweite der beiden Files das neuere ist:

```
rwx------ 5203/5200       0 Apr 13 15:18 1993 Beispiel/
rwx------ 5203/5200       0 Apr 13 15:17 1993 Beispiel/
Unterdirectory/
rw------- 5203/5200     198 Apr 13 15:17 1993 Beispiel/
Unterdirectory/drittes_file
rw------- 5203/5200     199 Apr 13 15:16 1993 Beispiel/erstes_file
rw------- 5203/5200     201 Apr 10 09:43 1993 Beispiel/zweites_file
rw------- 5203/5200    1675 Apr 10 08:43 1993 ein_file
rwx------ 5203/5200       0 Apr 13 15:18 1993 Beispiel/
rwx------ 5203/5200       0 Apr 18 09:58 1993 Beispiel/
Unterdirectory/
rw------- 5203/5200     232 Apr 18 09:58 1993 Beispiel/
Unterdirectory/drittes_file
```

Hierbei werden zusätzlich noch zu den Filenamen die Zugriffsrechte, die Kombination Userid/Groupid des Eigentümers, die Anzahl der Zeichen im File und die Veränderungsdaten aufgeführt. Da die Inhalte von Directory-Files nicht mit ins Archiv aufgenommen werden, wird die Größe von Directories oder Subdirectories immer mit 0 Bytes ausgewiesen.

4.3.5 Filesystem nach der Archivierung

Nach der erfolgreichen Archivierung des Filesystems in ein Archiv-File kann das Filesystem gelöscht werden. Dies geschieht mit dem Kommando **tar** nicht automatisch, sondern das Löschen muß explizit durchgeführt werden, zum Beispiel mit der Kommandofolge

```
rm -r Beispiel
rm ein_file
```

Das Kommando

```
ls -l
```

zur Auflistung der Files in der Working-Directory liefert danach am Bildschirm nur noch die folgende Ausgabe:

```
total 20
-rw-------  1 wgriege     10240 Apr 18 10:18 Beispiel.tar
drwx------  2 wgriege       512 Apr 10 08:37 Noch_eine_Directory
drwx------  2 wgriege       512 Apr 10 08:36 Weitere_Directory
```

Die Größe des Archiv-Files „Beispiel.tar“ hat sich nach dem Austausch des Files „drittes_file“ nicht geändert. Dies ist jedoch nicht der Regelfall. Im allgemeinen ändert sich die Größe eines Archiv-Files, wenn darin Files ausgetauscht werden. Da die Inhalte älterer Files im Archiv nicht gelöscht werden, wird das Archiv-File nach jedem Austausch und nach jeder Erweiterung in der Regel größer.

4.3.6 Extraktion eines Files oder eines Filesystems aus einem Archiv

Sollen Files mit dem Kommando **tar** aus einem bestehenden Archiv extrahiert werden, so muß die Grundoption **-x** gewählt werden.

Wiederherstellung des gesamten Archivs

Am einfachsten ist die Syntax des Kommandos, wenn alle im Archiv verzeichneten Files extrahiert werden sollen, also das gesamte Filesystem auf einmal wiederhergestellt werden soll.

Für den Fall des Archiv-Files „Beispiel.tar“ lautet das Kommando:

```
tar -xvf Beispiel.tar
```

Die Option **-f** dient wieder zur Extraktion aus dem Archiv-File „Beispiel.tar“, dessen Name als Parameter unmittelbar hinter dieser Option angegeben wird. Würde diese Option

nicht angegeben, so erfolgte die Extraktion aus einem Archiv auf einem Magnetband. Die Option **-v** bewirkt die folgende Bildschirmausgabe:

```
x Beispiel/
x Beispiel/Unterdirectory/
x Beispiel/Unterdirectory/drittes_file, 198 bytes, 1 blocks
x Beispiel/erstes_file, 199 bytes, 1 blocks
x Beispiel/zweites_file, 201 bytes, 1 blocks
x ein_file, 1675 bytes, 4 blocks
x Beispiel/Unterdirectory/drittes_file, 232 bytes, 1 blocks
```

Der Buchstabe „x“ in der ersten Spalte der Bildschirmausgabe deutet an, daß es sich bei dem Vorgang um eine Extraktion handelt.

Die Files werden demnach in der Reihenfolge wiederhergestellt, wie sie archiviert wurden, also wie sie im Archiv verzeichnet sind, und zwar ausgehend von der Working-Directory.

Damit stehen die Files mit den zugehörigen Directories erneut zur Verfügung, so daß mit ihnen weitergearbeitet werden kann. Das Kommando

ls -l

zur Auflistung der Files in der Working-Directory liefert am Bildschirm wieder:

```
total 28
drwx------  3 wgriege       512 Apr 19 22:56 Beispiel
-rw-------  1 wgriege     10240 Apr 18 10:18 Beispiel.tar
drwx------  2 wgriege       512 Apr 10 08:37 Noch_eine_Directory
drwx------  2 wgriege       512 Apr 10 08:36 Weitere_Directory
-rw-------  1 wgriege      1675 Apr 10 08:43 ein_file
```

Macht man nun mit dem Kommando

cd Beispiel/Unterdirectory

die Subdirectory „Beispiel/Unterdirectory“ zur Working-Directory, so erzeugt das Kommando

```
ls -l
```

zum Auflisten der Files in der Working-Directory am Bildschirm die Ausgabe:

```
total 4
-rw-------  1 wgriege       232 Apr 18 09:58 drittes_file
```

Es ist also nur die neueste Version des Files „drittes_file“ wiederhergestellt worden.

Mit dem Kommando **tar** kann auch auf ältere Versionen im Archiv zugegriffen werden. Das wird im nächsten Abschnitt beschrieben.

Extraktion einzelner Files aus einem Archiv

Um auch die Extraktion einzelner Files aus einem Archiv darstellen zu können, soll nun wieder der Zustand hergestellt werden, der unmittelbar vor der Wiederherstellung des gesamten Filesystems bestand. Es müssen demzufolge die Directory „Beispiel“ und das File „ein_file“ in der Home-Directory gelöscht werden. Dazu wird mit

```
cd $HOME
```

die Home-Directory zur Working-Directory gemacht und dann

```
rm -r Beispiel
rm ein_file
```

eingegeben. Durch das Kommando

```
tar -xvf Beispiel.tar
  Beispiel/Unterdirectory/drittes_file
```

wird das File „drittes_file“ aus der Subdirectory „Unterdirectory“ der Directory „Beispiel“ extrahiert. Die Option **-f** bedeutet wieder, daß in der Kommandozeile nach den Optionen als erstes der Name des Archiv-Files „Beispiel.tar“

angegeben werden muß. Die Option **-v** bewirkt folgende Bildschirmausgabe:

```
x Beispiel/
x Beispiel/Unterdirectory/
x Beispiel/Unterdirectory/drittes_file, 198 bytes, 1 blocks
x Beispiel/Unterdirectory/drittes_file, 232 bytes, 1 blocks
```

Es sind also die Directory „Beispiel“ und die Subdirectory „Unterdirectory“ angelegt worden. Das File „drittes_file“ ist in der neuesten Version in die Subdirectory „Unterdirectory“ kopiert worden. Jede Extraktion erfolgt immer ausgehend von der Working-Directory. Das erkennt man auch nach dem Kommando

```
ls -l
```

das am Bildschirm die folgenden Files anzeigt:

```
total 24
drwx------  3 wgriege       512 Apr 21 21:37 Beispiel
-rw-------  1 wgriege     10240 Apr 18 10:18 Beispiel.tar
drwx------  2 wgriege       512 Apr 10 08:37 Noch_eine_Directory
drwx------  2 wgriege       512 Apr 10 08:36 Weitere_Directory
```

Die Kommandofolge

```
cd Beispiel
ls -l
```

zeigt am Bildschirm jedoch nur noch:

```
total 4
drwx------  2 wgriege       512 Apr 21 21:37 Unterdirectory
```

und

```
cd Unterdirectory
ls -l
```

liefert am Bildschirm:

```
total 4
-rw-------  1 wgriege       232 Apr 18 09:58 drittes_file
```

Das File „drittes_file“ ist also in der Tat nur in der neuesten Version extrahiert worden.

Aus dem Archiv können auch in einer Kommandozeile mehrere Files, ausgehend von der Working-Directory, extrahiert werden. Zum Beispiel ist nach der Kommandofolge

```
cd $HOME
tar -xvf Beispiel.tar
                Beispiel/erstes_file
                   Beispiel/zweites_file
```

am Bildschirm

```
x Beispiel/erstes_file, 199 bytes, 1 blocks
x Beispiel/zweites_file, 201 bytes, 1 blocks
```

zu sehen. Das bedeutet, daß die beiden Files „erstes_file“ und „zweites_file“ in die Directory „Beispiel“ kopiert worden sind. Die Kommandofolge

```
cd Beispiel
ls -l
```

zeigt demnach am Bildschirm den Inhalt der Subdirectory „Beispiel“ an:

```
total 12
drwx------  2 wgriege       512 Apr 21 21:37 Unterdirectory
-rw-------  1 wgriege       199 Apr 13 15:16 erstes_file
-rw-------  1 wgriege       201 Apr 10 09:43 zweites_file
```

Als letztes kann nun noch das File „ein_file“ der Home-Directory aus dem Archiv-File „Beispiel.tar“ extrahiert werden:

```
cd $HOME
tar -xvf Beispiel.tar ein_file
```

Am Bildschirm erscheint danach die Meldung:

```
x ein_file, 1675 bytes, 4 blocks
```

Das File „ein_file“ ist also unmittelbar in die Working-Directory kopiert worden. Das Kommando

```
ls -l
```

zum Auflisten der Files in der Working-Directory liefert die Bildschirmausgabe:

```
total 28
drwx------  3 wgriege       512 Apr 21 22:36 Beispiel
-rw-------  1 wgriege     10240 Apr 18 10:18 Beispiel.tar
drwx------  2 wgriege       512 Apr 10 08:37 Noch_eine_Directory
drwx------  2 wgriege       512 Apr 10 08:36 Weitere_Directory
-rw-------  1 wgriege      1675 Apr 10 08:43 ein_file
```

Wird noch einmal das gleiche Kommando, nämlich

```
tar -xvf Beispiel.tar ein_file
```

eingegeben, so erscheint am Bildschirm wieder:

```
x ein_file, 1675 bytes, 4 blocks
```

Ein unter dem gleichen Namen „ein_file“ bereits vorhandenes File würde demzufolge von dem entsprechenden File aus dem Archiv überschrieben werden und wäre verloren.

Soll das versehentliche Überschreiben von Files weitgehend vermieden werden, so kann zusätzlich die Option **-w** verwendet werden. Diese bewirkt, daß **tar** die nächste Aktion anzeigt und auf Bestätigung wartet. Deshalb liefert nun das Kommando

```
tar -xvwf Beispiel.tar ein_file
```

am Bildschrim die folgende Anfrage:

```
x ein_file: rw------- 5203/5200    1675 Apr 10 08:43 1993 z
```

Der Cursor steht am Ende der Zeile, und das Kommando **tar** wartet auf eine Antwort. Wird nun ein **y** für „yes“ eingetippt und die Return-Taste gedrückt, so wird das File „ein_file“ extrahiert und eventuell ein unter dem gleichen Namen vorhandenes File überschrieben, und die nächste Aktion des Kommandos **tar** wird angezeigt. Jede mit Return abgeschlossene Eingabe, die nicht mit dem Buchstaben „y“ beginnt, bewirkt, daß das File „ein_file“ *nicht extrahiert* wird, und die nächste Aktion des Kommandos **tar** wird angezeigt. Auf diese Weise kann auch auf ältere Versionen eines Files im Archiv zugegriffen werden.

Das Archiv-File „Beispiel.tar“ ist auch weiterhin vorhanden und kann weiterverwendet werden. Da es aber seinen Zweck erfüllt hat, wird es gelöscht:

```
rm Beispiel.tar
```

Damit hat die Home-Directory wieder die ursprüngliche Gestalt angenommen. Das Kommando

```
ls -l
```

zeigt am Bildschirm nämlich wieder:

```
total 16
drwx------  3 wgriege       512 Apr 21 22:36 Beispiel
drwx------  2 wgriege       512 Apr 10 08:37 Noch_eine_Directory
drwx------  2 wgriege       512 Apr 10 08:36 Weitere_Directory
-rw-------  1 wgriege      1675 Apr 10 08:43 ein_file
```

Allgemeine Syntax

Die allgemeine Syntax des Kommandos **tar** zum Extrahieren von Files aus einem Archiv lautet:

```
tar -xvf filename.tar file1 file2 ...
                                ... filen
```

file1, *file2*, ... *filen* müssen dabei so angegeben werden, wie sie nach dem Kommando

```
tar -tf filename.tar
```

am Bildschirm erscheinen. Der Parameter *filename* kann jedoch ein absoluter oder relativer Pfadname sein.

Ausgangspunkt der Extraktion ist immer die Working-Directory. Muß zum Extrahieren eine Directory angelegt werden, so wird sie in der Working-Directory angelegt. Die Angabe von absoluten Pfadnamen von zu extrahierenden Files ist nicht möglich. Sind mehrere Files gleichen Namens in einem Archiv verzeichnet, so werden zwar alle Files extrahiert, aber das zuletzt archivierte überschreibt alle vorhergehenden. Wird zusätzlich die Option **-w** verwendet, so können auch ältere Versionen eines Files im Archiv extrahiert werden. Bereits vorhandene Files werden von Files gleichen Namens im Archiv ohne Warnung überschrieben.

4.3.7 Grundsätzliches zu den Optionen des Kommandos tar

Im allgemeinen werden die Optionen des Kommandos **tar** *nicht mit einem Minuszeichen* geschrieben.

Das hat zur Folge, daß die allgemeine Syntax zum Erzeugen eines neuen Archivs, zum Erweitern eines Archivs mit neuen Files, zum Anzeigen aller Files in einem Archiv, zum Austauschen von Files in einem Archiv und zum Extrahieren von Files aus einem Archiv auch folgendermaßen aussehen kann:

```
tar cvf filename.tar filename

tar rvf filename.tar file1 file2 ...
                                ... filen

tar tf filename.tar

tar uvf filename.tar file1 file2 ...
                                ... filen
```

```
tar xvf filename.tar file1 file2 ...
                                ... filen
```

Diese Form ist sogar die allgemein übliche. Sie ist in diesem Buch nur deshalb nicht gewählt worden, um den Unterschied zwischen Optionen, die unter dem Betriebssystem Unix in der Regel durch ein Minuszeichen gekennzeichnet sind, und Parametern, zum Beispiel Filenamen, des behandelten Kommandos zu verdeutlichen.

Alle Unix-Derivate erlauben beide Arten der Notation.

Weitere wählbare Optionen des Kommandos **tar** und Besonderheiten unter einigen Unix-Derivaten entnehme man dem Anhang „**tar** zum Nachschlagen“.

4.4 Weitere Kommandos zur Archivierung

Unter dem Betriebssystem Unix gibt es noch weitere Kommandos oder Utilities, die auch eine Archivierung von Files und Filesystemen bewirken können. Zum einen sind diese Kommandos Bestandteile der meisten Unix-Derivate, zum anderen stehen sie als Public-Domain-Produkte zur Verfügung.

Da alle diese Kommandos und Utilities nicht vollständig aufgezählt werden können, seien hier nur beispielhaft **gtar**, **ar** und **mdtar** etwas näher beschrieben.

4.4.1 GNU gtar

Das von der Free Software Foundation bereitgestellte Kommando GNU **gtar** ist eine Weiterentwicklung des Standard-**tar**-Kommandos. Es besitzt einige Grundoptionen, die bei **tar** nicht zur Verfügung stehen. Mit **gtar** ist ein Vergleich der Inhalte zweier Archive möglich, ohne daß die Filesysteme restauriert werden müssen. Weiter können mehrere Archive zu einem großen Archiv-File zusammengefaßt werden. Das Unix-Kommando **cat** ist dazu nicht geeignet, da es eine Änderung der internen Strukturen des neuen Archiv-

Files nicht berücksichtigen kann. Mit **gtar** ist es auch möglich, einzelne Files in einem Archiv zu löschen.

Genaueres zu den Grundoptionen entnehme man dem Anhang „**tar** zum Nachschlagen“.

Aufgrund der erweiterten Funktionalität gegenüber **tar** ist **gtar** sicherlich für eine Archivierung vorzuziehen. Leider ist nur **tar** standardmäßig im Betriebssystem Unix integriert. **gtar** müßte zusätzlich installiert werden oder **tar** ersetzen.

4.4.2 Aufbau von Bibliotheken: ar

Das Unix-Kommando **ar** wird in der Regel zum Aufbau von Bibliotheken benutzt, die vom Link-Editor verwendet werden sollen. Es ist wohl der Ursprung aller Kommandos oder Utilities zur Archivierung. Das Kommando **ar** besitzt jedoch nicht die Funktionsvielfalt von **tar**, kann aber trotzdem durchaus auch für allgemeine Archivierungszwecke verwendet werden. Allerdings ist **ar** für die Archivierung von Softwareprodukten oder Dokumentation in der Regel *nicht* gebräuchlich. Genaueres zum Unix-Kommando **ar** entnehme man den Man-Pages.

4.4.3 Mehrfache Archivierung: mdtar

Das Unix-Kommando **mdtar** ist in der Lage, mehrere Files gleichzeitig in verschiedene Archive zu schreiben. Ansonsten besitzt es keine über das Kommando **tar** hinausgehenden Funktionalitäten. Das Kommando **mdtar** wird in der Regel *nicht* für die Archivierung und Bereitstellung von Softwareprodukten verwendet. Es steht auch nicht mehr unter allen Unix-Derivaten zur Verfügung. Genaueres zum Unix-Kommando **mdtar** entnehme man den Man-Pages.

KAPITEL 5 `shar` - zur Zusammenfassung von Files in einem Shellscript

Filesysteme, die über Rechnernetze kopiert werden sollen, dürfen während des Kopiervorgangs nicht derart verändert werden, daß sie auf dem Zielrechner nicht in den ursprünglichen Zustand zurückversetzt werden können. Das Kommando **`tar`**, das ein Archiv-File erzeugt, legt in diesem File auch Zeichen ab, die zum Beispiel von Mailern, über die die elektronische Kommunikation geführt wird, so verändert werden können, daß sie später nicht mehr restaurierbar sind. Aus diesem Grund wurde eine Software entwickelt, die nur 7-Bit-ASCII-Zeichen zusätzlich zu den Daten in ein Archiv-File schreibt, die von Mailern nicht verändert werden.

Bei dieser Software handelt es sich um die Utility **`shar`**. Sie ist *nicht standardmäßig* unter dem Betriebssystem Unix vorhanden.

5.1 Verwendbare Software

Die Utility **`shar`** ist ein Programm, das in der Programmiersprache C geschrieben und als Public-Domain-Produkt verfügbar ist. Es gibt jedoch auch Shellscript-Ausgaben.

Von der Utility **`shar`** existieren sehr viele Varianten. Im folgenden wird eine Variante beschrieben, die von Gary Perlman, Wang Institute, stammt.

5.2 Zusammenfassung von Files in einem Shellscript

Die Funktionalität der Utility **`shar`** soll nun wieder anhand des im Kapitel „Anwendungsbeispiel“ erläuterten Beispiels demonstriert werden.

Die Kommandofolge

```
cd $HOME
ls -l
```

liefert am Bildschirm die Ausgabe:

```
total 16
drwx------  3 wgriege       512 Apr 13 15:18 Beispiel
drwx------  2 wgriege       512 Apr 10 08:37 Noch_eine_Directory
drwx------  2 wgriege       512 Apr 10 08:36 Weitere_Directory
-rw-------  1 wgriege      1675 Apr 10 08:43 ein_file
```

5.2.1 Erzeugung eines einfachen Shellscript-Archivs

Soll nun das Filesystem „Beispiel“ mit der Utility **shar** archiviert werden, so kann folgender Aufruf verwendet werden:

```
shar Beispiel
```

Auf dem Bildschirm erscheint danach die folgende Ausgabe:

```
#! /bin/sh
# This is a shell archive, meaning:
# 1. Remove everything above the #! /bin/sh line.
# 2. Save the resulting text in a file.
# 3. Execute the file with /bin/sh (not csh) to create the files:
#       Beispiel
# This archive created: Sun May 16 08:41:58 1993
export PATH; PATH=/bin:$PATH
if test ! -d 'Beispiel'
then
        mkdir 'Beispiel'
fi
cd 'Beispiel'
if test ! -d 'Unterdirectory'
then
        mkdir 'Unterdirectory'
fi
cd 'Unterdirectory'
if test -f 'drittes_file'
then
        echo shar: will not over-write existing file "'drittes_file'"
else
cat << \SHAR_EOF > 'drittes_file'
# Dieses ist der Inhalt des ersten Files in der Subdirectory
# "Unterdirectory".
# Der Text ist f"ur das folgende irrelevant.
# Er dient nur zur Illustration.

# Damit endet das File "drittes_file".
SHAR_EOF
fi # end of overwriting check
cd ..
if test -f 'erstes_file'
then
        echo shar: will not over-write existing file "'erstes_file'"
else
cat << \SHAR_EOF > 'erstes_file'
# Dieses ist der Inhalt des ersten Files in der Directory
# "Anwendungsbeispiel".
# Der Text ist f"ur das folgende irrelevant.
# Er dient nur zur Illustration.

# Damit endet das File "erstes_file".
SHAR_EOF
fi # end of overwriting check
if test -f 'zweites_file'
then
        echo shar: will not over-write existing file "'zweites_file'"
else
cat << \SHAR_EOF > 'zweites_file'
# Dieses ist der Inhalt des zweiten Files in der Directory
# "Anwendungsbeispiel".
# Der Text ist f"ur das folgende irrelevant.
# Er dient nur zur Illustration.

# Damit endet das File "zweites_file".
SHAR_EOF
fi # end of overwriting check
cd ..
#       End of shell archive
exit 0
```

Die Utility **shar** erzeugt demzufolge kein neues File, insbesondere auch kein Archiv-File, sondern schreibt standardmäßig das Shellscript-Archiv auf die Standardausgabe, also auf den Bildschirm.

Die Bildschirmausgabe läßt sich selbstverständlich leicht umlenken, nämlich durch folgendes Kommando zum Beispiel in das File „Beispiel.shar“:

```
shar Beispiel > Beispiel.shar
```

Danach befindet sich der Inhalt der obigen Bildschirmausgabe in dem File „Beispiel.shar“. Dieses ist nun das *Shellscript-Archiv* für das Filesystem „Beispiel“.

Das Kommando

```
ls -l
```

zur Anzeige des Inhalts der Working-Directory erzeugt nun folgende Bildschirmausgabe:

```
total 20
drwx------  3 wgriege       512 Apr 13 15:18 Beispiel
-rw-------  1 wgriege      1592 May 16 08:41 Beispiel.shar
drwx------  2 wgriege       512 Apr 10 08:37 Noch_eine_Directory
drwx------  2 wgriege       512 Apr 10 08:36 Weitere_Directory
-rw-------  1 wgriege      1675 Apr 10 08:43 ein_file
```

Eine Erweiterung des schon bestehenden Shellscript-Archivs „Beispiel.shar“ durch ein weiteres File, zum Beispiel „ein_file“, mit Hilfe der Utility **shar** ist nicht möglich.

Um auch das File „ein_file“ in ein Shellscript-Archiv mit dem Namen „Beispiel+.shar“ zu übernehmen, hätte das Kommando

```
shar Beispiel ein_file >
                          Beispiel+.shar
```

eingegeben werden können.

5.2.2 Verwendung von Optionen

Die im vorigen Abschnitt demonstrierte Art der Archivierung eines Filesystems in einem Shellscript-Archiv ist mit der Utility **shar** die einfachste. In den meisten Fällen genügt dieses Verfahren.

Wenn jedoch bei der Wiederherstellung des File-systems mehr Information gewünscht wird, so müssen zur Archivierung der Utility **shar** Optionen mitgegeben werden; **shar** kennt eine ganze Reihe davon. Eine vollständige Liste ist im Anhang „**shar** zum Nachschlagen“ aufgeführt.

Hier soll nur die Option **-a** vorgestellt werden. Die Archivierung geschieht wie bei der Verwendung von **shar** ohne eine Option. Die Ausgabe der Utility wird in ein File umgelenkt, zum Beispiel „Beispiel.opt.shar“:

```
shar -a Beispiel > Beispiel.opt.shar
```

Das Filesystem ist damit im Shellscript-Archiv „Beispiel.opt.shar“ archiviert worden, ohne daß die in dieser Form gebrauchte Utility eine Ausgabe am Bildschirm erzeugt.

In der Home-Directory zeigt das Kommando

```
ls -l
```

nun folgenden Inhalt an:

```
total 24
drwx------  3 wgriege       512 Apr 13 15:18 Beispiel
-rw-------  1 wgriege      2514 May 19 22:13 Beispiel.opt.shar
-rw-------  1 wgriege      1592 May 16 08:41 Beispiel.shar
drwx------  2 wgriege       512 Apr 10 08:37 Noch_eine_Directory
drwx------  2 wgriege       512 Apr 10 08:36 Weitere_Directory
-rw-------  1 wgriege      1675 Apr 10 08:43 ein_file
```

Wie anhand der angezeigten Zahl der Zeichen in den Files erkennbar ist, ist das neue Shellscript-Archiv „Beispiel.opt.shar“ deutlich größer als das vorher erzeugte „Bei-

spiel.shar“. Läßt man sich den Inhalt des Files „Beispiel.opt.shar“ mit dem Kommando

```
cat Beispiel.opt.shar
```

am Bildschirm anzeigen, so erhält man folgende Ausgabe:

```
#! /bin/sh
# This is a shell archive, meaning:
# 1. Remove everything above the #! /bin/sh line.
# 2. Save the resulting text in a file.
# 3. Execute the file with /bin/sh (not csh) to create the files:
#       Beispiel
# This archive created: Wed May 19 22:15:18 1993
export PATH; PATH=/bin:$PATH
if test ! -d 'Beispiel'
then
        echo shar: creating directory "'Beispiel'"
        mkdir 'Beispiel'
fi
echo shar: entering directory "'Beispiel'"
cd 'Beispiel'
if test ! -d 'Unterdirectory'
then
        echo shar: creating directory "'Unterdirectory'"
        mkdir 'Unterdirectory'
fi
echo shar: entering directory "'Unterdirectory'"
cd 'Unterdirectory'
echo shar: extracting "'drittes_file'" '(199 characters)'
if test -f 'drittes_file'
then
        echo shar: will not over-write existing file "'drittes_file'"
else
sed 's/^        X//' << \SHAR_EOF > 'drittes_file'
        X# Dieses ist der Inhalt des ersten Files in der Subdirectory
        X# "Unterdirectory".
        X# Der Text ist f"ur das folgende irrelevant.
        X# Er dient nur zur Illustration.
        X
        X# Damit endet das File "drittes_file".
SHAR_EOF
if test 199 -ne "`wc -c < 'drittes_file'`"
then
        echo shar: error transmitting "'drittes_file'" '(should have
been 199 characters)'
fi
fi # end of overwriting check
echo shar: done with directory "'Unterdirectory'"
cd ..
echo shar: extracting "'erstes_file'" '(199 characters)'
if test -f 'erstes_file'
then
        echo shar: will not over-write existing file "'erstes_file'"
else
sed 's/^      X//' << \SHAR_EOF > 'erstes_file'
```

```
        X# Dieses ist der Inhalt des ersten Files in der Directory
        X# "Beispiel".
        X# Der Text ist f"ur das folgende irrelevant.
        X# Er dient nur zur Illustration.
        X
        X# Damit endet das File "erstes_file".
SHAR_EOF
if test 199 -ne "`wc -c < 'erstes_file'`"
then
        echo shar: error transmitting "'erstes_file'" '(should have
been 199 characters)'
fi
fi # end of overwriting check
echo shar: extracting "'zweites_file'" '(201 characters)'
if test -f 'zweites_file'
then
        echo shar: will not over-write existing file "'zweites_file'"
else
sed 's/^        X//' << \SHAR_EOF > 'zweites_file'
        X# Dieses ist der Inhalt des zweiten Files in der Directory
        X# "Beispiel".
        X# Der Text ist f"ur das folgende irrelevant.
        X# Er dient nur zur Illustration.
        X
        X# Damit endet das File "zweites_file".
SHAR_EOF
if test 201 -ne "`wc -c < 'zweites_file'`"
then
        echo shar: error transmitting "'zweites_file'" '(should have
been 201 characters)'
fi
fi # end of overwriting check
echo shar: done with directory "'Beispiel'"
cd ..
#       End of shell archive
exit 0
```

Das Filesystem „Beispiel“ hat nun seinen Zweck erfüllt und kann mit dem Kommando

```
rm -r Beispiel
```

wieder gelöscht werden. Zur Wiederherstellung kann bei Bedarf das Shellscript-Archiv „Beispiel.shar“ oder das Shellscript-Archiv „Beispiel.opt.shar“ verwendet werden.

5.2.3 Zum Inhalt eines Shellscript-Archivs

In einem Shellscript-Archiv sind Bourne-Shell-Kommandos derart aufgeführt, daß sie bei der Ausführung des Shellscripts in der Bourne-Shell das ursprünglich vorhandene Filesystem wiederherstellen. Es sind ausschließlich Bourne-

Shell-Kommandos gewählt, da jedes Unix-Derivat zumindest mit der Bourne-Shell ausgeliefert wird. In der Regel werden nur die Kommandos **cat**, **sed**, **mkdir**, **cd** und einige wenige andere verwendet. Wird **shar** ohne Option aufgerufen, so wird für das Anlegen von Files das Unix-Kommando **cat** verwendet. Mit der Option **-a** werden anstelle von **cat** die Eigenschaften des Stream-Editors **sed** ausgenutzt.

Den genaueren Aufbau entnehme man dem obigen Shellscript-Archiv „Beispiel.shar“ oder dem Shellscript-Archiv „Beispiel.opt.shar“.

5.2.4 Zur Nomenklatur

Die Utility **shar** läßt als Namen des Shellscript-Archivs jeden unter Unix gültigen Filenamen zu. Trotzdem hat es sich eingebürgert, für den Namen des Shellscript-Archivs denjenigen des zu archivierenden Filesystems zu wählen und an ihn die Endung **.shar** anzuhängen. Dieser Brauch soll auch im folgenden beibehalten werden. Nicht ganz so häufig findet man für den Namen des Shellscript-Archivs auch die Endung **.sh** vor.

Damit sieht die allgemeine Syntax des Kommandos zum Erzeugen eines Shellscript-Archivs folgendermaßen aus:

```
shar -optionen filename >
                          filename.shar
```

filename ist dann durch den Namen der Directory des zu archivierenden Filesystems zu ersetzen. Er kann als absoluter oder relativer Pfadname angegeben werden. Im Shellscript-Archiv wird er so gespeichert, wie er angegeben wurde. Als **-***optionen* können die im Anhang „**shar** zum Nachschlagen“ aufgeführten Optionen der Utility **shar** eingesetzt werden.

Es ist auch möglich, mehrere Files oder Files zusammen mit Filesystemen in einem Shellscript-Archiv zu archivieren. In diesem Fall muß der Parameter *filename* durch die Namen der Files oder File-systeme ersetzt werden.

5.3 Extraktion aller Files aus einem Shellscript-Archiv

Wie das mit der Utility **shar** archivierte Filesystem wiederhergestellt werden kann, ist in den ersten Zeilen des zugehörigen Shellscript-Archivs beschrieben. Da im Shellscript-Archiv nur Bourne-Shell-Kommandos enthalten sind, kann die Extraktion auch nur in einer Bourne-Shell erfolgen. Zum Aufruf der neuen Bourne-Shell muß demzufolge das Unix-Kommando **sh** verwendet werden, dem der Name des Shellscript-Archivs als Parameter übergeben wird.

Als Alternative kann natürlich auch eine Bourne-Shell als Subshell eröffnet und das Shellscript-Archiv als Shellscript ausgeführt werden. In diesem Fall muß jedoch dem Shellscript-Archiv noch das Execute-Recht zugewiesen werden.

Die Wiederherstellung des mit **shar** archivierten Filesystems „Beispiel“ soll nun anhand der beiden oben erzeugten Shellscript-Archive „Beispiel.shar“ und „Beispiel.opt.shar“ demonstriert werden. In der Home-Directory ist das Filesystem „Beispiel“ gelöscht. Das Kommando

```
ls -l
```

zeigt unter anderem die beiden Shellscript-Archive an:

```
total 20
-rw-------  1 wgriege      2514 May 19 22:13 Beispiel.opt.shar
-rw-------  1 wgriege      1592 May 16 08:41 Beispiel.shar
drwx------  2 wgriege       512 Apr 10 08:37 Noch_eine_Directory
drwx------  2 wgriege       512 Apr 10 08:36 Weitere_Directory
-rw-------  1 wgriege      1675 Apr 10 08:43 ein_file
```

5.3.1 Extraktion aus dem einfachen Shellscript-Archiv

Das mit der Utility **shar** ohne die Angabe von Optionen erzeugte Shellscript-Archiv „Beispiel.shar“ kann nun wieder zur Extraktion aller Files aus dem Shellscript-Archiv herangezogen werden.

Dazu wird das Kommando

```
sh Beispiel.shar
```

eingegeben. Am Bildschirm erscheint daraufhin keine Meldung. Erst das Kommando

```
ls -l
```

zeigt am Bildschirm an, daß das Filesystem „Beispiel“ wiederhergestellt wurde:

```
total 24
drwx------  3 wgriege       512 May 22 09:23 Beispiel
-rw-------  1 wgriege      2514 May 19 22:13 Beispiel.opt.shar
-rw-------  1 wgriege      1592 May 16 08:41 Beispiel.shar
drwx------  2 wgriege       512 Apr 10 08:37 Noch_eine_Directory
drwx------  2 wgriege       512 Apr 10 08:36 Weitere_Directory
-rw-------  1 wgriege      1675 Apr 10 08:43 ein_file
```

Alle wiederhergestellten Files tragen jedoch das aktuelle Datum als Veränderungsdatum. Ursprünglich im Filesystem vorhandene Veränderungsdaten werden demzufolge nicht wieder eingesetzt. Das Shellscript-Archiv „Beispiel.shar“ ist unverändert geblieben. Es kann auch weiterhin verwendet werden. Da nun noch die Extraktion aus dem mit Optionen erzeugten Shellscript-Archiv demonstriert werden soll, wird das Filesystem „Beispiel“ wieder gelöscht:

```
rm -r Beispiel
```

5.3.2 Extraktion aus dem mit Optionen erzeugten Shellscript-Archiv

Das mit der Utility **shar** und der Option **-a** erzeugte Shellscript-Archiv „Beispiel.opt.shar“ kann jetzt ebenfalls zur Extraktion aller Files aus dem Shellscript-Archiv herangezogen werden.

Dazu wird das Kommando

```
sh Beispiel.opt.shar
```

eingegeben. Am Bildschirm erscheinen daraufhin die Meldungen:

```
shar: creating directory 'Beispiel'
shar: entering directory 'Beispiel'
shar: creating directory 'Unterdirectory'
shar: entering directory 'Unterdirectory'
shar: extracting 'drittes_file' (199 characters)
shar: done with directory 'Unterdirectory'
shar: extracting 'erstes_file' (199 characters)
shar: extracting 'zweites_file' (201 characters)
shar: done with directory 'Beispiel'
```

Diese Meldungen wurden schon bei der Erzeugung des Shellscript-Archivs aufgrund der Option **-a** unter Zuhilfenahme des Unix-Kommandos **echo** im Shellscript-Archiv eingetragen.

Das Filesystem „Beispiel“ ist wiederhergestellt worden, was nach der Eingabe des Kommandos

```
ls -l
```

am Bildschirm abzulesen ist:

```
total 24
drwx------  3 wgriege       512 May 22 09:44 Beispiel
-rw-------  1 wgriege      2514 May 19 22:13 Beispiel.opt.shar
-rw-------  1 wgriege      1592 May 16 08:41 Beispiel.shar
drwx------  2 wgriege       512 Apr 10 08:37 Noch_eine_Directory
drwx------  2 wgriege       512 Apr 10 08:36 Weitere_Directory
-rw-------  1 wgriege      1675 Apr 10 08:43 ein_file
```

Alle wiederhergestellten Files tragen das aktuelle Datum als Veränderungsdatum. Ursprünglich im Filesystem vorhandene Veränderungsdaten werden nicht wieder eingesetzt. Das Shellscript-Archiv „Beispiel.opt.shar“ ist unverändert geblieben. Es kann auch weiterhin verwendet werden. Da jedoch nun beide Shellscript-Archive ihren Zweck erfüllt haben, werden sie wieder gelöscht:

```
rm Beispiel.shar
rm Beispiel.opt.shar
```

5.3.3 Allgemeine Syntax

Zur Extraktion eines Filesystems aus einem Shellscript-Archiv wird folgendes Kommando eingegeben:

sh *filename*.**shar**

Für *filename*.**shar** ist dabei der Name des Shellscript-Archivs einzusetzen.

Als Alternative kann eine Bourne-Shell als Subshell eröffnet werden:

```
sh
chmod u+x filename.shar
filename.shar
C-d
```

Durch das letzte Kommando **C-d** (Control-d) wird die Bourne-Shell nach der Extraktion wieder verlassen. Für *filename*.**shar** ist dabei der Name des Shellscript-Archivs einzusetzen.

Aus einem Shellscript-Archiv kann nur das gesamte Filesystem wiederhergestellt werden. Die Extraktion einzelner Files ist ohne Manipulation am Shellscript-Archiv nicht möglich. Als Veränderungsdatum der Files wird immer das aktuelle Datum eingesetzt.

Ein schon vorhandenes File, das den gleichen Namen trägt wie eines, das aus einem Shellscript-Archiv extrahiert werden soll, wird während der Extraktion des gesamten Filesystems *nicht* überschrieben, und eine entsprechende Warnung wird am Bildschirm ausgegeben.

KAPITEL 6 `compress` - zur Komprimierung von Files `uuencode` - zur Kodierung von Files

Files zu komprimieren scheint nach den bisherigen Erläuterungen in den vorangegangenen Kapiteln überflüssig zu sein. Die Komprimierung gewinnt jedoch an Bedeutung, wenn Files über stark belastete Rechnernetze transportiert werden sollen. Je größer ein File ist, desto länger dauert nämlich der Transport und entsprechend länger wird das betreffende Netz blockiert.

In den folgenden Abschnitten werden zunächst Verfahren zur Komprimierung von Files dargestellt, danach wird die Anwendung dieser Verfahren in verschiedenen Unix-Kommandos und Utilities gezeigt. Ausführlich wird die Verwendung des Unix-Kommandos **`compress`** behandelt.

Um Files über Rechnernetze fehlerfrei transportieren zu können, müssen diese Files manchmal noch kodiert werden, damit die Daten nicht verfälscht werden. Dafür wird das Unix-Kommando **`uuencode`** erläutert.

6.1 Komprimierung von Files

Im folgenden werden die Aufgaben einer Komprimierung und einzelne Kommandos oder Utilities zur Komprimierung von Files vorgestellt. Die Ausführungen sollen einen ersten Eindruck der Möglichkeiten bringen. Sie erheben keinen Anspruch auf Vollständigkeit.

6.1.1 Aufgaben einer Komprimierung

Die Komprimierung eines Files dient dazu, die Zahl der im File vorhandenen Zeichen möglichst klein zu halten, ohne die Information, die sich in diesem File befindet, zu verän-

dern. Je kleiner die Zahl der Zeichen in einem File ist, desto schneller läßt sich dieses File über Rechnernetze kopieren, da diese Zeit proportional zur Größe des Files, also zur vorhandenen Anzahl der Zeichen im File ist. Da zunehmend Rechnernetze für den Transport von Files benutzt werden, gewinnt die Komprimierung von Files immer mehr an Bedeutung.

6.1.2 Komprimierungsverfahren

Im folgenden sollen einige gängige Komprimierungsverfahren vorgestellt werden. Sie beruhen im allgemeinen darauf, Zeichen oder ganze Zeichenketten zu neuen Zeichen zusammenzufassen und diese anstelle der ursprünglichen zu speichern.

Die unten erläuterten Verfahren stammen aus den „Frequently Asked Questions (FAQ)“ der beiden Usenet-Newsgroups „comp.compression“ und „comp.compression.research“, die regelmäßig von Jean-loup Gailly veröffentlicht werden. Viele Verfahren unterliegen Patentrechten, so daß sie nicht ohne weiteres benutzt werden dürfen.

Auf Einzelheiten wird in den Erläuterungen nicht eingegangen. Sie entnehme man den FAQ oder der dort zitierten Originalliteratur. Es gibt kein „bestes Komprimierungsverfahren“, da jedes bestimmte Annahmen über das Vorkommen von Zeichen in einem File machen muß, um optimal zu komprimieren. Diese Voraussetzungen sind für ein gegebenes File in der Regel selten vollständig erfüllt.

Huffman-Komprimierung

Die Huffman-Komprimierung beruht auf der Tatsache, daß die Häufigkeit der einzelnen Zeichen in einem File unterschiedllich ist. Entsprechend der Häufigkeit werden den Zeichen neue Kodierungen zugewiesen. Optimal ist das Verfahren, wenn jedes Zeichen mit einer Wahrscheinlichkeit

vorkommt, die sich mathematisch als ganzzahlige Potenz von 1/2 darstellen läßt.

Ein davon abgeleitetes Verfahren ist die Shannon-Fano-Kodierung.

Arithmetische Komprimierung

Bei der arithmetischen Komprimierung werden ganze Zeichenketten entsprechend ihrer Häufigkeit in einem File einem Bitstring zugeordnet, der jedoch unter Umständen sehr lang werden kann.

Im Gegensatz zur Huffman-Komprimierung können die Zeichen mit beliebiger Wahrscheinlichkeit auftreten, das Verfahren bleibt optimal. Nachteilig wirkt sich jedoch vielfach der große CPU-Zeit-Verbrauch aus, der erforderlich ist, um die Bitstrings zu berechnen.

Ziv-Lempel-Komprimierung

Unter der Ziv-Lempel-Komprimierung versteht man zwei Verfahren, die in den Jahren 1977 und 1978 von Jakob Ziv und Abraham Lempel vorgestellt wurden. Diese Art der Komprimierung vermeidet den hohen CPU-Zeit-Verbrauch der arithmetischen Komprimierung dadurch, daß Zeichenketten in ein „Wörterbuch“ geschrieben werden und auf dieses Bezug genommen wird. Das Unix-Kommando **`compress`** und die Utility GNU **`gzip`** verwenden jeweils eines dieser Verfahren.

Vergleich der Komprimierungsverfahren

In den FAQ ist auch ein Vergleich der in den verschiedenen Komprimierungskommandos oder Utilities verwendeten Komprimierungsverfahren veröffentlicht. Die Ergebnisse hängen selbstverständlich von dem zu komprimierenden File ab.

6.1.3 Komprimierung von Files mit `compress`

Zum Komprimieren von Files steht unter den Unix-Derivaten standardmäßig das Kommando **`compress`** zur Verfügung, dessen Verwendung zur Komprimierung im folgenden näher beschrieben werden soll. Es nutzt ein Verfahren der Ziv-Lempel-Komprimierung aus.

Komprimierung eines Files

Die Komprimierung soll an dem File „ein_file“ demonstriert werden, das sich in der Home-Directory der Userid „wgriege“ befindet und im Kapitel „Anwendungsbeispiel“ näher erläutert ist.

Die Kommandofolge

```
cd $HOME
ls -l
```

liefert am Bildschirm die Ausgabe:

```
total 16
drwx------  3 wgriege       512 May 22 09:44 Beispiel
drwx------  2 wgriege       512 Apr 10 08:37 Noch_eine_Directory
drwx------  2 wgriege       512 Apr 10 08:36 Weitere_Directory
-rw-------  1 wgriege      1675 Apr 10 08:43 ein_file
```

Das Kommando zum Komprimieren des Files „ein_file“ lautet einfach:

```
compress ein_file
```

Das Kommando zum Auflisten der Files in der Working-Directory:

```
ls -l
```

zeigt danach am Bildschirm folgendes an:

```
total 16
drwx------  3 wgriege       512 May 22 09:44 Beispiel
drwx------  2 wgriege       512 Apr 10 08:37 Noch_eine_Directory
drwx------  2 wgriege       512 Apr 10 08:36 Weitere_Directory
-rw-------  1 wgriege       828 Apr 10 08:43 ein_file.Z
```

Die Zahl der Zeichen im komprimierten File, das an der Endung **.Z** zu erkennen ist, ist demzufolge gegenüber dem nicht komprimierten File etwa halbiert worden. Das ursprüngliche File ist gelöscht.

Zugriffsrechte und das Veränderungsdatum des ursprünglichen Files sind auf das komprimierte File übertragen worden.

Zum Inhalt eines komprimierten Files

Der Inhalt des Files „ein_file.Z“ läßt sich weder mit dem Unix-Kommando **`cat`** noch mit dem Unix-Kommando **`more`** zufriedenstellend am Bildschirm anzeigen, da in der Regel der vollständige 8-Bit-Zeichensatz verwendet wird, der auch Steuersequenzen für die Bildschirmdarstellung enthält. Dadurch ergibt sich häufig eine Verfälschung der Anzeige.

Nur mit Hilfe eines Editors, zum Beispiel mit GNU **`emacs`**, läßt sich der Inhalt des komprimierten Files darstellen, da der Editor alle Zeichen im File lediglich als Zeichen interpretiert und anzeigt und keines als Steueranweisung für die Bildschirmausgabe.

Soll jedoch der Inhalt eines komprimierten Files in der ursprünglichen Form, also dekomprimiert am Bildschirm angezeigt werden, ohne das File explizit zu dekomprimieren, so kann dafür das Unix-Kommando **`zcat`** verwendet werden.

```
zcat ein_file.Z
```

liefert am Bildschirm die folgende Ausgabe, die mit der Anzeige des ursprünglichen Files „ein_file“ im Kapitel „Anwendungsbeispiel“ übereinstimmt:

```
# In diesem File stehen einige v"ollig belanglose Daten. Es dient
# lediglich zum F"ullen der Beschreibung.

Anhand dieses Beispiels soll sp"ater die Komprimierung von Files
demonstriert werden. Deshalb braucht der folgende Text nicht
gelesen zu werden.

abc defg hijkl mnopqr stuvwxy zab cdef ghijk lmnopr stuvwx yz 12 34
567 890 eins zwei drei vier f"unf sechs sieben acht neun zehn elf
zw"olf dreizehn

abc defg hijkl mnopqr stuvwxy zab cdef ghijk lmnopr stuvwx yz 12 34
567 890 eins zwei drei vier f"unf sechs sieben acht neun zehn elf
zw"olf dreizehn

abc defg hijkl mnopqr stuvwxy zab cdef ghijk lmnopr stuvwx yz 12 34
567 890 eins zwei drei vier f"unf sechs sieben acht neun zehn elf
zw"olf dreizehn

abc defg hijkl mnopqr stuvwxy zab cdef ghijk lmnopr stuvwx yz 12 34
567 890 eins zwei drei vier f"unf sechs sieben acht neun zehn elf
zw"olf dreizehn

abc defg hijkl mnopqr stuvwxy zab cdef ghijk lmnopr stuvwx yz 12 34
567 890 eins zwei drei vier f"unf sechs sieben acht neun zehn elf
zw"olf dreizehn

abc defg hijkl mnopqr stuvwxy zab cdef ghijk lmnopr stuvwx yz 12 34
567 890 eins zwei drei vier f"unf sechs sieben acht neun zehn elf
zw"olf dreizehn

abc defg hijkl mnopqr stuvwxy zab cdef ghijk lmnopr stuvwx yz 12 34
567 890 eins zwei drei vier f"unf sechs sieben acht neun zehn elf
zw"olf dreizehn

abc defg hijkl mnopqr stuvwxy zab cdef ghijk lmnopr stuvwx yz 12 34
567 890 eins zwei drei vier f"unf sechs sieben acht neun zehn elf
zw"olf dreizehn

abc defg hijkl mnopqr stuvwxy zab cdef ghijk lmnopr stuvwx yz 12 34
567 890 eins zwei drei vier f"unf sechs sieben acht neun zehn elf
zw"olf dreizehn

01234567890123

# Das ist nun das Ende des Files "ein_file".
```

Allgemeine Syntax

Die allgemeine Syntax zum Komprimieren des Files *filename* lautet:

compress *filename*

Danach wird ein File mit dem Namen *filename*.**Z** erzeugt, das über die gleichen Zugriffsrechte verfügt und das gleiche Veränderungsdatum trägt wie das File *filename*. Das File *filename* ist gelöscht worden.

Anstelle von *filename* können auch mehrere Namen von Files *file1*, *file2*, ..., *filen* angegeben werden, die dann nacheinander komprimiert werden.

Es ist nicht möglich, anstelle von *filename* den Namen einer Subdirectory einzusetzen. Wird das Kommando in der obigen Form eingegeben, so wird ein File dann *nicht komprimiert*, wenn das komprimierte File größer würde als das nicht komprimierte.

Dem Kommando **compress** können auch Optionen mitgegeben werden. Diese sind im Anhang „**compress** und **uuencode** zum Nachschlagen“ erläutert.

6.1.4 Dekomprimierung komprimierter Files

Mit **compress** komprimierte Files können selbstverständlich wieder dekomprimiert, also in ihren ursprünglichen Zustand zurückversetzt werden.

Dekomprimierung eines mit compress komprimierten Files

Das mit dem Unix-Kommando **compress** komprimierte File „ein_file.Z“ kann mit Hilfe des Kommandos **compress** auch wieder dekomprimiert werden. Es muß lediglich noch die Option **-d** angegeben werden:

```
compress -d ein_file.Z
```

Das Kommando

```
ls -l
```

zum Anzeigen des Inhalts der Working-Directory zeigt danach am Bildschirm wieder folgendes an:

```
total 16
drwx------  3 wgriege       512 May 22 09:44 Beispiel
drwx------  2 wgriege       512 Apr 10 08:37 Noch_eine_Directory
drwx------  2 wgriege       512 Apr 10 08:36 Weitere_Directory
-rw-------  1 wgriege      1675 Apr 10 08:43 ein_file
```

Das File „ein_file.Z“ ist also wieder in das ursprüngliche File „ein_file“ zurückverwandelt worden. Das File „ein_file.Z“ ist gelöscht.

Allgemeine Syntax

Das mit dem Unix-Kommando **compress** komprimierte File *filename*.**Z** wird folgendermaßen wieder dekomprimiert:

compress -d *filename*.**Z**

Alternativ kann auch folgende Syntax verwendet werden:

uncompress *filename*.**Z**

Beide Fälle sind äquivalent. Das ursprüngliche File *filename* wird wiederhergestellt. Das File *filename*.**Z** wird gelöscht.

Anstelle von *filename*.**Z** können auch mehrere Namen von komprimierten Files *file1*.**Z**, *file2*.**Z**, ..., *filen*.**Z** angegeben werden, die dann nacheinander dekomprimiert werden.

6.1.5 Weitere Kommandos und Utilities zur Komprimierung

compress ist nicht das einzige Kommando unter dem Betriebssystem Unix, das Files komprimieren kann. Es existieren auch Utilities, also Programme, die nicht im Betriebssystem Unix verankert sind, die Files komprimieren können, und zwar häufig mit besseren Ergebnissen als die Kommandos liefern würden.

Weitere Kommandos zur Komprimierung

Standardmäßig stehen unter Unix noch zwei weitere Kommandos zur Komprimierung von Files zur Verfügung, nämlich **compact** und **pack**, wobei eingeschränkt werden muß, daß **compact** nicht mehr unter allen Unix-Derivaten verfügbar ist. Beide nutzen jeweils ein Verfahren der Huffman-Komprimierung aus.

Die allgemeine Syntax zum Komprimieren des Files *filename* mit **compact** lautet:

compact *filename*

Danach wird das komprimierte File *filename*.**C** erzeugt und das File *filename* gelöscht. Dekomprimiert wird das File *filename*.**C** durch:

uncompact *filename*.**C**

Die allgemeine Syntax zum Komprimieren des Files *filename* mit **pack** lautet:

```
pack filename
```

Danach wird das komprimierte File *filename*.**z** erzeugt und das File *filename* gelöscht. Dekomprimiert wird das File *filename*.**z** durch:

```
unpack filename.z
```

Genaueres zu diesen beiden Kommandos lese man in den Man-Pages nach, die durch

```
man compact
man pack
```

aufgerufen werden können.

Diese Kommandos werden eigentlich nur noch aus Kompatibilitätsgründen unter Unix unterstützt, damit Files, die früher mit diesen Kommandos komprimiert wurden, auch wieder dekomprimiert werden können. Das Kommando **compress** erzielt in der Regel eine bessere Komprimierung als **compact** oder **pack**.

Utilities zur Komprimierung

Die bisher vorgestellten Kommandos liefern im allgemeinen schlechte Ergebnisse bei der Komprimierung von Binär-Files, zum Beispiel Grafikdateien, da diese in der Regel eine scheinbar zufällige Verteilung der Bits im File enthalten.

Zur Komprimierung von Binär-Files sind deshalb besondere Komprimierungsverfahren entwickelt worden. Diese sind in den „Frequently Asked Questions (FAQ)“ der Usenet-Newsgroups „comp.compression“ und „comp.compression.research“ erläutert. Entsprechende Utilities zur Komprimierung solcher Files sind dort ebenfalls angegeben. Sie sollen hier nicht näher erörtert werden, da sie den Rahmen dieser Einführung sprengen würden.

Ein Verfahren zur Komprimierung allgemeiner Files wurde von der Free Software Foundation in die Utility GNU **gzip** integriert. Da diese frei verfügbar ist, wenn das mitgelieferte

Copyright nicht verändert wird, soll diese Utility kurz vorgestellt werden.

Soll das File „ein_file“, das im Kapitel „Anwendungsbeispiel“ beschrieben ist, mit **gzip** komprimiert werden, so muß folgendes eingegeben werden:

```
gzip ein_file
```

Danach entsteht ein neues File. Das Kommando zum Anzeigen des Inhalts der Working-Directory

```
ls -l
```

zeigt am Bildschirm folgendes an:

```
total 16
drwx------  3 wgriege       512 May 22 09:44 Beispiel
drwx------  2 wgriege       512 Apr 10 08:37 Noch_eine_Directory
drwx------  2 wgriege       512 Apr 10 08:36 Weitere_Directory
-rw-------  1 wgriege       343 Apr 10 08:43 ein_file.z
```

Es ist demzufolge das komprimierte File „ein_file.z“ entstanden, das ursprüngliche File „ein_file“ ist gelöscht worden. Die Zugriffsrechte und das Veränderungsdatum sind erhalten geblieben. Diese Eigenschaften entsprechen denen des Kommandos **compress**.

Anhand der angezeigten Größe des Files „ein_file.z“, 343 Bytes, ist erkennbar, daß **gzip** im Vergleich zu **compress**, das das File „ein_file.Z“ mit 828 von ursprünglich 1675 Bytes erzeugt hat, ein deutlich besseres Komprimierungsergebnis liefert.

Allgemein wird das File *filename* mit **gzip** durch den Aufruf

gzip *filename*

komprimiert. Es entsteht das File *filename*.**z** und das ursprüngliche File *filename* ist gelöscht. Zugriffsrechte und das Veränderungsdatum bleiben erhalten.

Da **gzip** genauso wie **pack** die Endung **.z** verwendet, kann es unter Umständen zu Konflikten kommen, wenn mit **unpack** ein mit **gzip** komprimiertes File dekomprimiert werden soll. Umgekehrt ist aber **gzip** in der Lage, auch mit **compress** oder mit **pack** komprimierte Files wieder zu dekomprimieren. Um dieser Problematik auszuweichen, wird von neueren Versionen der Utility GNU **gzip**, die jedoch noch nicht überall die älteren ersetzt haben, die Endung **.gz** erzeugt.

Das File *filename*.**z** kann mit **gzip** durch den Aufruf

gzip -d *filename*.**z**

wieder dekomprimiert werden. Das ursprüngliche File *filename* wird wiederhergestellt. Alternativ kann auch folgender Aufruf zur Dekomprimierung verwendet werden:

gunzip *filename*.**z**

Entsprechendes gilt für komprimierte Files *filename*.**gz**, die mit neueren Versionen der Utility GNU **gzip** erzeugt wurden.

Genaueres zu den Utilities **gzip** oder **gunzip** entnehme man der mit der Software ausgelieferten Dokumentation.

6.2 Kodierung von Files

Alle bisher vorgestellten Verfahren zur Archivierung oder Komprimierung von Files können bewirken, daß neue Files angelegt werden.

In diesen neuen Files wird der 8-Bit-Zeichensatz in der Regel vollständig ausgenutzt, das heißt, insbesondere ist auch vielfach das achte Bit im Zeichen besetzt, also nicht durch 0, sondern durch 1 dargestellt.

Einige Übertragungsnetze haben jedoch die Eigenschaft, das achte Bit im Zeichen zu verändern, so daß beim Transport dieser Files über solche Netze Information, die ursprünglich in den Files vorhanden ist, verlorengeht.

6.2.1 Aufgabe einer Kodierung

Die Aufgabe einer Kodierung ist es nun, aus einem File, in dem Zeichen enthalten sind, bei denen alle acht Bits besetzt sein können, ein neues zu erzeugen, in dem jedes achte Bit eines Zeichens durch 0 dargestellt wird, ohne daß der Informationsgehalt, der ursprünglich im File vorhanden war, verringert wird.

Solch ein Kommando zur Kodierung von Files ist standardmäßig unter Unix vorgesehen.

6.2.2 Kommandos zur Kodierung von Files

Das unter Unix standardmäßig verfügbare Kommando zur Kodierung von Files ist **`uuencode`**. Mit dem Unix-Kommando **`uudecode`** werden solche kodierten Files wieder in ihren ursprünglichen Zustand zurückversetzt, also dekodiert.

Kodierung eines Files mit `uuencode`

Das Unix-Kommando **`uuencode`** bewirkt, daß aus drei Bytes im File vier Bytes erzeugt werden, wobei in jedem dieser vier Bytes nur sechs Bits von 0 verschieden sein können, die erzeugten Zeichen am Bildschirm jedoch lesbar und auch druckbar sind.

Anhand des mit der Utility **`gzip`** erzeugten komprimierten Files „ein_file.z“ soll nun der Aufruf des Kommandos **`uuencode`** erläutert werden.

Das File „ein_file.z“ wird mit **`uuencode`** folgendermaßen kodiert:

```
uuencode ein_file.z ein_file.z.neu >
                              ein_file.z.uu
```

Der erste Parameter des Kommandos **`uuencode`** ist demzufolge der Name des Files „ein_file.z“, das kodiert werden soll. Der zweite Parameter ist der Name des Files „ein_file.z.neu“, in das nach der Dekodierung das *dekodierte* File hineingeschrieben werden soll. Wird nur ein Parameter angegeben, so erwartet **`uuencode`** die Eingabe von der Standardeingabe, und dieser einzige Parameter wird als Name für das später wieder dekodierte File angenommen.

Die Ausgabe des Kommandos **uuencode** wird durch das Umlenksymbol **>** in das File „ein_file.z.uu“ geschrieben. Es enthält also die kodierte Form des Files „ein_file.z“. Ohne die Umlenkung würde die Ausgabe auf den Bildschirm als Standardausgabe geschrieben.

Das Kommando

```
ls -l
```

zeigt nun am Bildschirm folgendes an:

```
total 20
drwx------  3 wgriege       512 May 22 09:44 Beispiel
drwx------  2 wgriege       512 Apr 10 08:37 Noch_eine_Directory
drwx------  2 wgriege       512 Apr 10 08:36 Weitere_Directory
-rw-------  1 wgriege       343 Apr 10 08:43 ein_file.z
-rw-------  1 wgriege       507 Jun  5 11:12 ein_file.z.uu
```

Es ist also wie erwartet das File „ein_file.z.uu“ erzeugt worden, das die kodierte Form des Files „ein_file.z“ enthält. Das ursprüngliche File ist dabei nicht verändert worden. Da bei der Kodierung aus drei Bytes vier neue Bytes gemacht werden und im kodierten File auch noch Kontrollinformationen enthalten sind, ist das kodierte File „ein_file.z.uu“ etwas mehr als ein Drittel größer als das unkodierte File „ein_file.z“.

Den genauen Inhalt des Files zeigt das Kommando

```
cat ein_file.z.uu
```

am Bildschirm an:

```
begin 600 ein_file.z.neu
M'XL( '1LQBL ^V5R6[#, Q$[ J*@7,/LG4[MD@*%+WV7L@6+;&5Y52TL CK
M2Z7(1Q3(C=!HAD\\4#.\)3@FHOZO' DR4* $XL2><*CZ&-FCIFB3C[T0MG:@
M-,=.BBT-9H9(CGWD)F :-:4:8]0$1QDO)$W(Q/68 -R8YQ1L<G M1$66/5,4
MB#:![["M-SD7%>] M,W=,68TX].G")L91UR<9L@H#CI1=(=F2!!MKU-F.31@N
MC=L^>DJ.\$&G 4G9!N-)(Q1L&J]68VS=Z/W6( #7=T27^OU/UAF,A^/IC,G6
M:%2&+S)BD:\JSA.6*ZPWYN[^ 8]/BS(RP70DAM,GX\"%HQI3"Z$FZ"N9:FUO
M"V.B44$HZ*!C:Z:CCKF]V,K9#>N&=<.Z8?T_K,5RM=YHKL9J9?1OV%H!BQK5
<Y[3>E:7L=/M?%CHJ;?[9:EG-S2_8)VP:BP8 ,5R

end
```

Alle Zeichen sind demnach am Bildschirm erkennbar und auch druckbar. In der ersten Zeile ist noch der nach der Dekomprimierung zu wählende Filename „ein_file.z.neu“ eingetragen. Anfang und Ende des kodierten Teils sind durch „begin“ bzw. „end“ gekennzeichnet.

Die Zahl 600, die in die erste Zeile eingetragen ist, ist die oktale Darstellung der Zugriffsrechte, die vom File „ein_file.z“ übernommen wurden und bei der Dekodierung dem File „ein_file.z.neu“ übertragen werden. Die Ziffer „6“ beschreibt die Zugriffsrechte des Eigentümers des Files, hier also die Summe aus dem Lese-recht mit der Ziffer „2“ und dem Schreibrecht mit der Ziffer „4“. Die Zugriffsrechte der Gruppe und der Sonstigen sind hier in beiden Fällen durch die Ziffer „0“ dargestellt. Sie erhalten demzufolge keinerlei Zugriffsrechte.

Allgemein lautet also die Syntax für den Aufruf des Kommandos **uuencode** zur Kodierung des Files *filename*:

```
uuencode filename dekod_file > filename.uu
```

Für *dekod_file* ist der Name des Files einzusetzen, das nach der Dekodierung den dekodierten Code enthalten soll. Es hat sich eingebürgert, für den Namen des kodierten Files denjenigen des zu kodierenden Files zu wählen und die Endung **.uu** anzuhängen, damit erkennbar wird, daß es mit **uuencode** kodiert wurde. Manchmal wird auch für das kodierte File die Endung **.uue** verwendet.

Läßt man *filename* in der Aufrufzeile weg, so wird die Eingabe von der Standardeingabe erwartet. Wird die Ausgabe nicht in das File *filename*.**uu** umgelenkt, so wird die Ausgabe auf die Standardausgabe geschrieben. Damit ist das Kommando **uuencode** auch für eine Pipe verwendbar.

Das File *filename*.**uu** kann nun unbeschadet über Rechnernetze transportiert oder kopiert werden.

Dekodierung eines Files mit uudecode

Wird nun ein mit **uuencode** kodiertes File über Rechnernetze transportiert, so kann es vorkommen, daß an den

Anfang des Files Daten eingefügt werden oder auch an das Ende angehängt werden. Insbesondere haben Mailer, die die elektronische Kommunikation vermitteln, diese Eigenschaft.

Aus dem File „ein_file.z.uu“ kann dadurch das File „ein_file.z.uu.mail“ geworden sein, dessen Inhalt das Kommando

```
cat ein_file.z.uu.mail
```

am Bildschirm anzeigt:

```
An diese Stelle kann ein Mailer irgendetwas hineingeschrieben haben.

begin 600 ein_file.z.neu
M'XL( '1LQBL ^V5R6[#, Q$[ J*@7,/LG4[MD@*%+WV7L@6+;&5Y52TL CK
M2Z7(1Q3(C=!HAD\\4#.\)3@FHOZO' DR4* $XL2><*CZ&-FCIFB3C[T0MG:@
M-,=.BBT-9H9(CGWD)F :-:4:8]0$1QDO)$W(Q/68 -R8YQ1L<G M1$66/5,4
MB#:![["M-SD7%>] M,W=,68TX].G")L91UR<9L@H#CI1=(=F2!!MKU-F.31@N
MC=L^>DJ.\$&G 4G9!N-)(Q1L&J]68VS=Z/W6( #7=T27^OU/UAF,A^/IC,G6
M:%2&+S)BD:\JSA.6*ZPWYN[^ 8]/BS(RP70DAM,GX\"%HQI3"Z$FZ"N9:FUO
M"V.B44$HZ*!C:Z:CCKF]V,K9#>N&=<.Z8?T K,5RM=YHKL9J9?1OV%H!BQK5
<Y[3>E:7L=/M?%CHJ;?[9:EG-S2 8)VP:BP8 ,5R

end

Und an diese Stelle kann etwas angef"ugt worden sein.
```

Dieses so veränderte File kann nun trotzdem dekodiert werden, da im kodierten Teil, also zwischen „begin“ und „end“, nichts verändert wurde.

Zum Dekodieren wird das Unix-Kommando **uudecode** verwendet:

```
uudecode ein_file.z.uu.mail
```

Am Bildschirm wird daraufhin *keine* Meldung angezeigt. Erst das Kommando

```
ls -l
```

zeigt am Bildschirm das neu entstandene File an:

```
total 28
drwx------  3 wgriege       512 May 22 09:44 Beispiel
drwx------  2 wgriege       512 Apr 10 08:37 Noch_eine_Directory
drwx------  2 wgriege       512 Apr 10 08:36 Weitere_Directory
-rw-------  1 wgriege       343 Apr 10 08:43 ein_file.z
-rw-------  1 wgriege       343 Jun  6 08:46 ein_file.z.neu
-rw-------  1 wgriege       507 Jun  5 11:12 ein_file.z.uu
-rw-------  1 wgriege       632 Jun  6 08:30 ein_file.z.uu.mail
```

Das durch die Dekodierung mit **uudecode** neu entstandene File ist „ein_file.z.neu“. Dieses ist also der Filename, der zur Kodierung mit **uuencode** angegeben wurde. Die anderen Files, insbesondere „ein_file.z.uu.mail“ und „ein_file.z.uu“, sind während der Dekodierung nicht verändert worden. Das ursprünglich zu kodierende File „ein_file.z“ ist ebenfalls noch vorhanden. Anhand der Zahl der Zeichen, die sich im File befinden, also 343, läßt sich schon vermuten, daß dieses mit dem neuen File „ein_file.z.neu“ übereinstimmt.

Zur Verifizierung dieser Vermutung kann das Unix-Kommando **diff** verwendet werden, das zwei Files miteinander vergleicht, inwieweit beide Inhalte übereinstimmen. Die Eingabe

```
diff ein_file.z ein_file.z.neu
```

erzeugt am Bildschirm *keine* Ausgabe. Das bedeutet, daß der Inhalt beider Files übereinstimmt und daß die Dekodierung wieder zum ursprünglichen File zurückgeführt hat. Würden die Files nicht übereinstimmen, so wären die Differenzen am Bildschirm angezeigt worden.

Die allgemeine Syntax zum Dekodieren des mit **uuencode** kodierten Files *filename* lautet demnach:

uudecode *filename*

Erzeugt wird dabei das dekodierte File, dessen Name bei der Kodierung festgelegt wurde. Auf die Endung **.uu** oder **.uue** ist hier verzichtet worden, da in der Regel nur erweiterte Files dekodiert werden.

Das Kommando **uudecode** ignoriert alle Einträge im zu dekodierenden File, die sich vor der Zeile, die mit „begin“ anfängt, und sich hinter der Zeile, die mit „end“ anfängt, befinden.

KAPITEL 7 Bereitstellung kopierbarer Software oder Dokumentation auf einem Rechner

In den Kapiteln „**tar** - zur Archivierung von Files und Filesystemen“, „**shar** - zur Zusammenfassung von Files in einem Shellscript“ und „**compress** - zur Komprimierung von Files, **uuencode** - zur Kodierung von Files“ sind alle diejenigen Unix-Kommandos behandelt worden, mit deren Hilfe es möglich ist, Software oder Dokumentation in einer nahezu standardisierten Form anderen zum Kopieren zur Verfügung zu stellen.

Für das Bereitstellen derart behandelter Software oder Dokumentation werden in der Regel sogenannte Anonymous-FTP-Server verwendet, die das Kopieren der Software oder Dokumentation auf andere Rechner ermöglichen.

In den folgenden Abschnitten soll ein Weg vorgeschlagen werden, der es ermöglicht, die Software oder Dokumentation auf einem Anonymous-FTP-Server bereitzustellen.

7.1 Voraussetzungen an die Software und Dokumentation

Die Software oder Dokumentation, die anderen Anwendern zur Verfügung gestellt werden soll, sollte einige Voraussetzungen erfüllen, damit sie nutzbringend angewendet werden kann.

Für die Bereitstellung von Dokumentation lassen sich wahrscheinlich keine allgemein gültigen Regeln aufstellen. Als selbstverständlich sollte jedoch gelten, daß die Beschreibungen, die in der Dokumentation enthalten sind, weitestgehend richtig, möglichst umfassend und vollständig in einem eigenen File oder Filesystem zusammgefaßt sind.

Für die bereitzustellende Software sollte als selbstverständlich gelten, daß sie bei der Anwendung keine Daten in unerwünschter Weise verändert oder gar löscht, wissentlich keine Viren, Würmer oder Trojanische Pferde enthält.

Da für den Einsatz von Software, im Gegensatz zu einer Dokumentation, besondere Anforderungen gestellt werden, sollten noch einige Voraussetzungen erfüllt sein. Diese Voraussetzungen sollen nun kurz dargestellt werden.

7.1.1 Die Software ist in einem eigenen Filesystem enthalten

Für die Software sollte ein eigenes Filesystem geschaffen sein, das nur Files oder Subdirectories enthält, die diese Software betreffen. Das hat den Vorteil, daß später bei der Wiederherstellung des Filesystems keine Kollisionen mit bereits vorhandenen Filenamen in der Working-Directory auftreten. Der Name des Filesystems sollte zudem auf den Namen der Software hindeuten.

7.1.2 Die Software enthält eine ausführliche Dokumentation

Im Filesystem sollte eine eigene Subdirectory enthalten sein, die den Namen „doc“ trägt und in der die Software ausführlich dokumentiert ist. Die Dokumentation sollte zum einen die vorzunehmenden Installationsschritte beschreiben und zum anderen den Gebrauch der Software erläutern. Nützlich ist es auf jeden Fall, auf Probleme, die bei der Installation der Software unter speziellen Unix-Derivaten auftreten können, hinzuweisen. Für die Benutzung der Software ist die Verwendung einer Man-Page hilfreich und damit deren Bereitstellung erwünscht.

7.1.3 Die Software enthält ein komplettes Makefile

In dem Filesystem ist ein komplettes Makefile enthalten, das nicht nur alle Anweisungen zur Kompilierung der Programme enthält, sondern mit

```
make install
```

auch eine vollständige Installation der Software ermöglicht. Die im Makefile vorzunehmenden Anpassungen an ein lokales Rechnersystem sollten ausführlich beschrieben sein.

7.2 Beispiel einer Bereitstellung von Software oder Dokumentation

Der Weg, der zur Bereitstellung von Software oder Dokumentation auf einem Anonymous-FTP-Server eingeschlagen werden kann, soll nun anhand des im Kapitel „Anwendungsbeispiel“ beschriebenen Beispiels erläutert werden.

Dieses Anwendungsbeispiel erfüllt sicherlich *nicht* die Voraussetzungen des vorigen Abschnitts, da es dafür zu einfach gewählt ist, kann jedoch wohl gerade wegen der Einfachheit besser zur Verständlichkeit beitragen als ein komplettes, dafür aber komplexes Filesystem.

Das Filesystem, das auf einem Anonymous-FTP-Server bereitgestellt werden soll, ist also das Filesystem „Beispiel“. Die Kommandofolge

```
cd $HOME
ls -l
```

liefert am Bildschirm:

```
total 16
drwx------  3 wgriege       512 May 22 09:44 Beispiel
drwx------  2 wgriege       512 Apr 10 08:37 Noch_eine_Directory
drwx------  2 wgriege       512 Apr 10 08:36 Weitere_Directory
-rw-------  1 wgriege      1675 Apr 10 08:43 ein_file
```

7.3 Erstellung eines Archivs

Entsprechend den im Kapitel „**tar** - zur Archivierung von Files und Filesystemen“ vorgestellten Kommandos zur Archivierung eines Filesystems wird zunächst einmal aus dem Filesystem „Beispiel“ ein Archiv-File erzeugt.

Dazu gibt man zweckmäßigerweise das folgende Kommando ein:

```
tar -cf Beispiel.tar Beispiel
```

Im Kapitel „**tar** - zur Archivierung von Files und Filesystemen“ ist ausführlich beschrieben, daß danach das Archiv-File „Beispiel.tar“ erzeugt wird, das später die Wiederherstellung des Filesystems „Beispiel“ erlaubt.

Im ersten Schritt des Weges ist damit das Archiv-File „Beispiel.tar“ erzeugt worden.

Soll Dokumentation, die nur in einem einzigen File enthalten ist, bereitgestellt werden, so ist es empfehlenswert, diesen Schritt der Archivierung in ein Archiv-File zu überspringen.

7.4 Komprimierung des Archiv-Files

In der Regel ist das nun vorhandene Archiv-File, gerade wenn es aus vielen Softwarekomponenten besteht, oder das File, in dem sich die Dokumentation befindet, recht groß. Um solch ein File über stark beanspruchte Rechnernetze zu kopieren, müßte beim Übertragungsvorgang entsprechend lange gewartet werden, da die Kopierzeit proportional zur Größe des Files ist.

Um diese Zeitspanne möglichst klein zu halten, bietet es sich an, das Archiv-File oder das File, in dem sich die Dokumentation befindet, zu komprimieren. Die entsprechenden Kommandos sind im Kapitel „**compress** - zur Komprimierung von Files, **uuencode** - zur Kodierung von Files“ ausführlich beschrieben. Obwohl die Utility GNU **gzip** in der Regel bessere Komprimierungsergebnisse liefert als das Unix-Kommando **compress**, ist eine Verwendung von **compress** vorzuziehen, da dieses Kommando systembedingt unter allen Unix-Derivaten vorhanden ist, während **gzip** nicht überall installiert ist.

Das Archiv-File „Beispiel.tar“ sollte demnach folgendermaßen komprimiert werden:

```
compress Beispiel.tar
```

Danach steht das komprimierte Archiv-File „Beispiel.tar.Z“ zur Verfügung. Das Kommando

```
ls -l
```

zum Anzeigen des Inhalts der Working-Directory, in diesem Beispiel also der Home-Directory, liefert am Bildschirm die folgende Ausgabe:

```
total 20
drwx------  3 wgriege       512 May 22 09:44 Beispiel
-rw-------  1 wgriege       830 Jun  7 22:52 Beispiel.tar.Z
drwx------  2 wgriege       512 Apr 10 08:37 Noch_eine_Directory
drwx------  2 wgriege       512 Apr 10 08:36 Weitere_Directory
-rw-------  1 wgriege      1675 Apr 10 08:43 ein_file
```

Das komprimierte Archiv-File „Beispiel.tar.Z“ steht nun im zweiten Schritt des Weges zur Übernahme auf einem Anonymous-FTP-Server zur Verfügung.

Soll Dokumentation, die nur in einem einzigen File enthalten ist, bereitgestellt werden, so erhält der Filename in diesem Schritt entsprechend die Endung **.Z**.

7.5 Erstellung und Komprimierung des Archiv-Files

In den vorigen beiden Abschnitten ist die Erstellung und die Komprimierung des Archiv-Files in zwei Schritten beschrieben worden.

Wenn jedoch der Pipe-Mechanismus unter Unix zu Hilfe genommen wird, so kann die Erstellung und Komprimierung des Archiv-Files auch in einem Schritt durchgeführt werden.

7.5.1 Erstellung und Komprimierung des Archiv-Files in einem Schritt

Die Erstellung und Komprimierung des Archiv-Files in einem Schritt soll nun wieder anhand des Anwendungsbeispiels „Beispiel“ demonstriert werden.

Um das Filesystem „Beispiel“ in ein komprimiertes Archiv-File zu verwandeln, kann folgendes Kommando eingegeben werden:

```
tar -cf - Beispiel | compress >
                          Beispiel.tar.Z
```

Das Minuszeichen hinter den Optionen **-cf** des Kommandos **tar** bedeutet, daß das Ergebnis der Archivierung des Filesystems „Beispiel“ durch das Kommando **tar** auf die Standardausgabe geschrieben wird. Von dort liest es das Kommando **compress**, was durch das Pipe-Symbol | bewirkt wird. Standardmäßig würde dann **compress** die Ausgabe ebenfalls auf die Standardausgabe, also auf den Bildschirm, schreiben. Das wird durch die Umlenkung **>** in das File „Beispiel.tar.Z“ verhindert.

Das Kommando

```
ls -l
```

zeigt nun am Bildschirm folgendes an:

```
total 20
drwx------  3 wgriege       512 May 22 09:44 Beispiel
-rw-------  1 wgriege      1109 Jun 12 09:35 Beispiel.tar.Z
drwx------  2 wgriege       512 Apr 10 08:37 Noch_eine_Directory
drwx------  2 wgriege       512 Apr 10 08:36 Weitere_Directory
-rw-------  1 wgriege      1675 Apr 10 08:43 ein_file
```

Die Größe des so komprimierten Archiv-Files „Beispiel.tar.Z“, 1109 Bytes, stimmt nicht mit der Größe des in zwei Schritten archivierten und danach komprimierten Archiv-Files, 830 Bytes, überein.

Das ist dadurch zu erklären, daß das Kommando **tar** in der aufgerufenen Form zum Archivieren in ein File eine andere Blockgröße verwendet als beim Archivieren auf die Standardausgabe. Für die Wiederherstellung des gesamten Filesystems spielt das aber keine Rolle.

7.5.2 Allgemeine Syntax

Soll das Filesystem *filename* in einem Schritt archiviert und komprimiert werden, so lautet dafür die allgemeine Syntax des Kommandos:

```
tar -cf - filename | compress >
                              filename.tar.Z
```

Die Angabe des Minuszeichens hinter den beiden Optionen **-cf** des Kommandos **tar** bewirkt, daß **tar** den Inhalt des Archiv-Files auf die Standardausgabe schreibt. Das Kommando **compress** liest den Inhalt des Archiv-Files von der Standardausgabe und würde das komprimierte Archiv-File auch wieder auf die Standardausgabe schreiben, wenn sie nicht durch **>** in das File *filename***.tar.Z** umgelenkt würde.

7.6 Erstellung eines Shellscript-Archivs

Manchmal wird anstelle eines komprimierten Archiv-Files zur Bereitstellung von Software, die von anderen kopiert werden kann, auch ein Shellscript-Archiv verfügbar gemacht. Im Kapitel „**shar** - zur Zusammenfassung von Files in einem Shellscript“ ist die allgemeine Vorgehensweise zur Erstellung eines Shellscript-Archivs beschrieben worden.

Soll das Filesystem „Beispiel“ in ein Shellscript-Archiv umgewandelt werden, so kann das Kommando

```
shar Beispiel > Beispiel.shar
```

eingegeben werden. Im File „Beispiel.shar“ befindet sich nun das Shellscript-Archiv. Das Shellscript-Archiv kann nun seinerseits wiederum komprimiert werden:

```
compress Beispiel.shar
```

Daraufhin wird das File „Beispiel.shar.Z“ erzeugt, das das komprimierte Shellscript-Archiv für das File-system „Beispiel“ enthält.

Mit Hilfe des Pipe-Mechanismus können diese Schritte auch zu einem Schritt zusammengefaßt werden. Dazu kann man folgendes Kommando eingeben:

```
shar Beispiel | compress >
                        Beispiel.shar.Z
```

Die Utility **shar** schreibt dabei den Inhalt des Shellscript-Archivs auf die Standardausgabe, von wo es das Kommando **compress** liest. **compress** würde den Inhalt des komprimierten Shellscript-Archivs auch wieder auf die Standardausgabe schreiben, wenn sie nicht durch **>** in das File „Beispiel.shar.Z“ umgelenkt würde.

7.7 Bereitstellung auf einem Rechner

Vielerorts werden zur Bereitstellung von kopierbarer Software oder Dokumentation Anonymous-FTP-Server zur Verfügung gestellt.

Von diesen kann die Software oder Dokumentation in einfacher Weise auf andere Rechner kopiert werden.

7.7.1 Anonymous-FTP-Server

Viele Rechenzentren stellen auf einer Unix-Workstation oder einem anderen Rechner eine Userid mit dem Namen „anonymous“ zur Verfügung. Diese Userid besitzt nur eingeschränkte Rechte, nämlich nur genau die, die zum Kopieren von Files auf andere Rechner benötigt werden. Von anderen Rechnern kann auf die Userid „anonymous“ zugegriffen werden, ohne daß ein spezielles Password zur Authentifikation verwendet werden muß, um unter der Userid abgelegte Files zu kopieren.

Dadurch sind diese Anonymous-FTP-Server besonders ausgezeichnet, Software oder Dokumentation anderen Anwendern oder Rechnern zum Kopieren zur Verfügung zu stellen. Der Zugriff auf einen Anonymous-FTP-Server von einem anderen Rechner aus wird im Kapitel „Weiterbehandlung

kopierter oder empfangener Software oder Dokumentation“ ausführlich beschrieben.

7.7.2 Bereitstellung der Software oder Dokumentation auf einem Anonymous-FTP-Server

In der Regel hat ein Anwender auf einem Rechner, auf dem auch ein Anonymous-FTP-Server installiert ist, keine Schreibrechte in die von der Userid „anonymous“ verwalteten Directories. Aus diesem Grund ist ein Kopieren von Files in diese Directories nur mit Hilfe des für den Anonymous-FTP-Server zuständigen Systemadministrators möglich.

Soll also Software oder Dokumentation über einen Anonymous-FTP-Server anderen Anwendern öffentlich zur Verfügung gestellt werden, so hat sich der Eigentümer der Software oder Dokumentation zunächst einmal an den für den Anonymous-FTP-Server zuständigen Systemadministrator zu wenden. Dieser entscheidet in der Regel, ob und in welche Directory der Userid „anonymous“ die Software kopiert werden kann.

In dem vorgestellten Beispiel wird danach der für den Anonymous-FTP-Server zuständige Systemadmini-strator das File „Beispiel.tar.Z“, das File „Beispiel.shar“, das File „Beispiel.shar.Z“ oder ein anderes komprimiertes File, das nur Dokumentation enthält, in eine Directory der Userid „anonymous“ kopieren.

Damit ist die Software oder Dokumentation zum Kopieren auf andere Rechner bereitgestellt.

KAPITEL 8 Versendung kopierbarer Software oder Dokumentation

In den Kapiteln „**tar** - zur Archivierung von Files und Filesystemen“, „**shar** - zur Zusammenfassung von Files in einem Shellscript“ und „**compress** - zur Komprimierung von Files, **uuencode** - zur Kodierung von Files“ sind alle diejenigen Unix-Kommandos behandelt worden, mit deren Hilfe es möglich ist, Software oder Dokumentation in einer nahezu standardisierten Form anderen zuzuschicken.

Für das Zuschicken können im allgemeinen zwei Wege gewählt werden. Zum einen ist es das Mail-System, über das die elektronische Kommunikation geführt wird. Damit kann ein oder können mehrere Empfänger gezielt erreicht werden. Zum anderen ist es das Usenet-Newssystem, bei dem sogenannte Artikel, die auch Software oder Dokumentation enthalten können, in Newsgroups gesendet werden, auf die dann auch andere Leser zugreifen können. Hierbei ist dann nicht mehr erkennbar, wer die Software oder Dokumentation nutzt.

In den folgenden Abschnitten sollen diese Wege näher erläutert werden.

Die Software oder Dokumentation, die zum Versenden bestimmt ist, sollte dieselben Voraussetzungen erfüllen wie die für das Kopieren auf einem Rechner bereitgestellte. Neben den selbstverständlichen Voraussetzungen, die im Kapitel „Bereitstellung kopierbarer Software oder Dokumentation auf einem Rechner“ erwähnt sind, gelten für die Versendung von Software folgende Regeln:

1. Die Software ist in einem eigenen Filesystem enthalten.
2. Die Software enthält eine ausführliche Dokumentation.

3. Die Software enthält ein komplettes Makefile.

8.1 Beispiel einer Versendung

Die Wege, die zur Versendung von Software oder Dokumentation eingeschlagen werden können, sollen nun anhand des im Kapitel „Anwendungsbeispiel“ beschriebenen Beispiels erläutert werden. Dieses Anwendungsbeispiel erfüllt sicherlich *nicht* die Voraussetzungen des vorigen Abschnitts, da es dafür zu einfach gewählt ist, kann jedoch wohl gerade wegen der Einfachheit besser zur Verständlichkeit beitragen als ein komplettes, dafür aber komplexes Filesystem.

Das Filesystem, das versendet werden soll, ist also das Filesystem „Beispiel“. Die Kommandofolge

```
cd $HOME
ls -l
```

liefert am Bildschirm die Ausgabe:

```
total 16
drwx------  3 wgriege       512 May 22 09:44 Beispiel
drwx------  2 wgriege       512 Apr 10 08:37 Noch_eine_Directory
drwx------  2 wgriege       512 Apr 10 08:36 Weitere_Directory
-rw-------  1 wgriege      1675 Apr 10 08:43 ein_file
```

8.2 Erzeugung eines Shellscript-Archivs

Entsprechend der im Kapitel „**shar** - zur Zusammenfassung von Files in einem Shellscript“ vorgestellten Utility zur Archivierung eines Filesystems wird zunächst einmal aus dem Filesystem „Beispiel“ ein Shellscript-Archiv erzeugt.

Dazu gibt man zweckmäßigerweise folgendes ein:

```
shar Beispiel > Beispiel.shar
```

Im Kapitel „**shar** - zur Zusammenfassung von Files in einem Shellscript“ ist ausführlich beschrieben, daß damit das Shellscript-Archiv „Beispiel.shar“ erzeugt wird, das später die Wiederherstellung des Filesystems „Beispiel“ erlaubt.

Als erster Schritt ist damit das Shellscript-Archiv „Beispiel.shar“ erzeugt.

Es ist nicht üblich, Shellscript-Archive zu komprimieren, die versendet werden sollen. Deshalb wird an dieser Stelle die Komprimierung nicht weiter behandelt.

Soll Dokumentation versendet werden, die sich nur in einem einzigen File befindet, so kann dieser Schritt der Archivierung selbstverständlich übersprungen werden.

8.3 Versendung über Mail oder News

Zur Versendung von Software oder Dokumentation über Netze wird die elektronische Kommunikation verwendet. Dafür stehen mehrere Verfahren zur Verfügung. Zwei davon, nämlich das Mail-System und das Usenet-Newssystem, sollen im folgenden näher beschrieben werden. Vorausgesetzt werden allerdings Grundkenntnisse in der Verwendung dieser Systeme.

Wenn sich in einem File, das im Shellscript-Archiv enthalten ist, auch Zeichen befinden, bei denen sich das achte Bit von 0 unterscheidet, also zum Beispiel Files, die ausführbare Programme enthalten, oder Grafik-Files, so ist das gesamte Shellscript-Archiv als Binär-File zu behandeln. Die Vorgehensweise zur Versendung von Binär-Files wird weiter unten erläutert.

8.3.1 Versendung über Mail

Standardmäßig gibt es unter den Unix-Derivaten das Kommando **`mail`**, das zum Austausch elektronischer Post vorgesehen ist. Die ausführliche Beschreibung der Verwendung des Kommandos findet man wieder in den zugehörigen Man-Pages, die durch

```
man mail
```

aufgerufen und gelesen werden können.

Vielfach ist das Kommando **mail** durch anwenderfreundlichere Programme ersetzt. Als Beispiel sei hier die Utility **elm** erwähnt. In diesen Fällen ist in den folgenden Ausführungen das Kommando **mail** durch das andere Programm, also zum Beispiel durch **elm**, zu ersetzen.

Das im vorigen Abschnitt erzeugte File „Beispiel.shar“ kann dann auf folgende Weise einfach zum Beispiel an die Userid „wgriege@gwdg.de“ geschickt werden:

```
mail wgriege@gwdg.de < Beispiel.shar
```

Der Ausdruck „wgriege@gwdg.de“ ist dabei die vollständige Adresse der Userid „wgriege“ im Internet. Die Standardeingabe des Kommandos **mail** wird durch die Angabe des Umlenksymbols **<** aus dem File „Beispiel.shar“ entnommen. Damit erhält die Userid „wgriege@gwdg.de“ das File „Beispiel.shar“ als elektronische Post zugesandt.

Allgemein wird demzufolge das Shellscript-Archiv *filename*.**shar** der Userid *userid*@*nodeid* über **mail** folgendermaßen zugesandt:

mail *userid*@*nodeid* **<** *filename*.**shar**

Die Weiterbehandlung der elektronischen Post beim Empfänger wird im Kapitel „Weiterbehandlung kopierter oder empfangener Software oder Dokumentation“ erläutert.

Enthält das Filesystem *filename* ein Binär-File oder mehrere Binär-Files, also zum Beispiel ausführbare Programme, so muß das Shellscript-Archiv *filename*.**shar** vor der Versendung so behandelt werden, als ob es ein vollständiges Binär-File ist. Die vor der Versendung dann notwendige Kodierung ist im übernächsten Abschnitt beschrieben.

8.3.2 Versendung über News

Eine weitere Möglichkeit zur Versendung von Software ist die Nutzung des Usenet-Newssystems. Das Newssystem besteht aus einer großen Zahl weltweit operierender *Newsserver*, die Diskussionsbeiträge geordnet in sogenannten *Newsgroups* austauschen. Die Newsgroups sind nach Dis-

kussionsthemen gegliedert. Die Diskussionsbeiträge, die sogenannten *Artikel*, werden von *Newsreadern*, den Clients der Newsserver, an das Newssystem übergeben. Die Newsreader sind auf Rechnern installiert, die von Anwendern benutzt werden können.

Da es sehr viele verschiedene Newsreader gibt, kann an dieser Stelle kein allgemein gültiges Verfahren beschrieben werden, nach dem die Software versendet werden kann.

Grundsätzlich verwendet jeder Newsreader einen Editor, zum Beispiel den GNU **`emacs`**, mit dessen Hilfe die Software, also das File „Beispiel.shar“, an einen Artikel angehängt werden kann.

Enthält das über News zu versendende Filesystem ein oder mehrere Binär-Files, also zum Beispiel ausführbare Programme, so muß das Shellscript-Archiv vor der Versendung so behandelt werden, als ob es ein vollständiges Binär-File ist. Die vor der Versendung dann notwendige Kodierung ist im übernächsten Abschnitt beschrieben.

Wichtig ist die Wahl einer geeigneten Newsgroup, an die der Artikel mit der Software gerichtet wird. Genaueres dazu kann sicherlich jeder News-Admini-strator mitteilen.

Das Verfahren, das angewendet werden kann, um derart versendete Software beim Empfänger zu bearbeiten, wird im Kapitel „Weiterbehandlung kopierter oder empfangener Software oder Dokumentation“ erläutert.

8.4 Erzeugung und Versendung eines Shellscript-Archivs

Im vorigen Abschnitt wurde die Versendung eines Filesystems in zwei Schritten erläutert. Als erstes wurde dazu das Filesystem in einem Shellscript-Archiv archiviert, als zweites über **`mail`** oder News versendet. Dieses Verfahren kann auch in einem Schritt, also in einer Kommandozeile durchgeführt werden, wenn der Pipe-Mechanismus verwendet wird.

8.4.1 Erzeugung und Versendung in einem Schritt

Das Filesystem „Beispiel“, das im Kapitel „Anwendungsbeispiel“ erläutert ist, soll nun mit Hilfe des Pipe-Mechanismus unter Unix in einem Schritt der Userid „wgriege@gwdg.de“ zugesendet werden.

Das im folgenden angebene Verfahren verwendet das Kommando **`mail`**. Für die Versendung über News ist dabei kein allgemein gültiges Konzept angebbar, da nicht alle Newsreader innerhalb einer Pipe verwendet werden können.

Die Versendung über **`mail`** in einem Schritt geschieht folgendermaßen:

```
cd $HOME
shar Beispiel | mail wgriege@gwdg.de
```

Dabei wird die Eigenschaft der Utility **`shar`** ausgenutzt, daß das Shellscript-Archiv auf die Standardausgabe geschrieben wird.

8.4.2 Allgemeine Syntax

Das Filesystem *`filename`* wird über **`mail`** der Userid *`userid@nodeid`* in einem Schritt auf folgende Weise zugesandt:

`shar` *`filename`* | **`mail`** *`userid@nodeid`*

Die Utility **`shar`** schreibt dabei das Shellscript-Archiv auf die Standardausgabe. Durch das Pipesymbol | wird die Standardausgabe der Utility **`shar`** in die Standardeingabe des Kommandos **`mail`** geleitet.

8.5 Versendung eines Binär-Files

Auch Binär-Files, also zum Beispiel ausführbare Programme oder Grafik-Files, können versendet werden. Sie müssen jedoch vor der Versendung kodiert werden, damit die Systeme der elektronischen Kommunikation keine Veränderung der Daten vornehmen können, die sich beim Empfänger nicht mehr rückgängig machen lassen.

8.5.1 Kodierung eines Binär-Files

Die Kodierung eines Binär-Files wird nun anhand eines Grafik-Files dargestellt.

Die Kommandofolge

```
cd $HOME
ls -l
```

zeigt am Bildschirm ein neues File an:

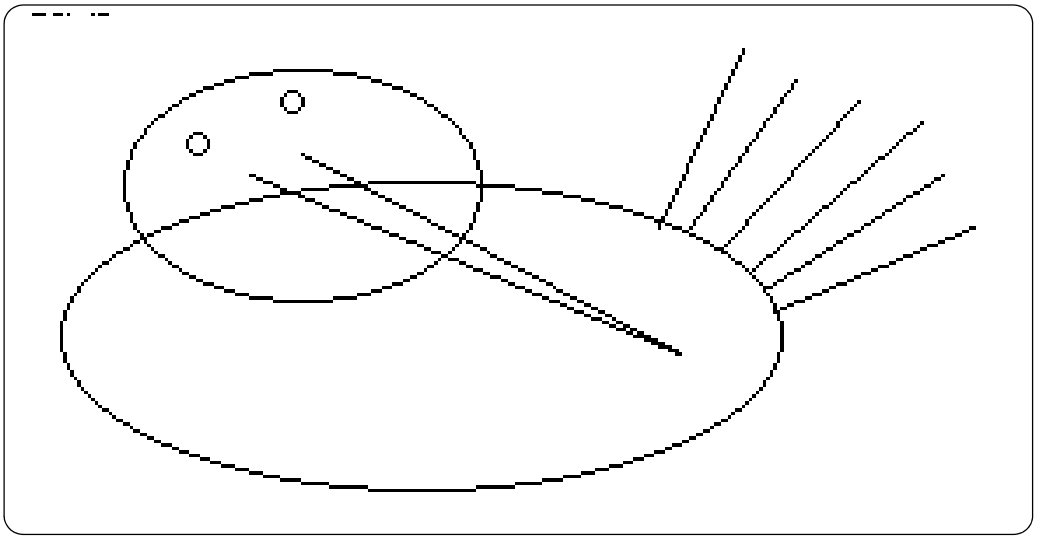
```
total 24
drwx------  3 wgriege       512 May 22 09:44 Beispiel
drwx------  2 wgriege       512 Apr 10 08:37 Noch_eine_Directory
drwx------  2 wgriege       512 Apr 10 08:36 Weitere_Directory
-rw-------  1 wgriege      1675 Apr 10 08:43 ein_file
-rw-------  1 wgriege      5438 Jun 14 22:41 vogel.tif
```

Es handelt sich also um das File „vogel.tif“, das eine Grafik im TIFF-Format enthält. Der grafische Inhalt des Files ist mit den Kommandos **cat** oder **more** am Bildschirm nicht anzeigbar. Erst ein Grafik-Viewer, zum Beispiel das Programm **xv**, kann den Inhalt des Files als Grafik auf einem grafikfähigen Bildschirm unter X-Windows darstellen.

In diesem Fall liefert der Aufruf

```
xv vogel.tif
```

am Bildschirm folgende Ausgabe:



Vor dem Versenden muß das Binär-File „vogel.tif“ kodiert werden. Das geschieht durch das Kommando

```
uuencode vogel.tif vogel.tif >
                              vogel.tif.uu
```

Im Kapitel „**compress** - zur Komprimierung von Files, **uuencode** - zur Kodierung von Files“ ist ausführlich beschrieben, daß sich die kodierte Form des Files „vogel.tif“ nun im File „vogel.tif.uu“ befindet. Das Kommando

```
ls -l
```

liefert nun am Bildschirm die Ausgabe:

```
total 32
drwx------  3 wgriege       512 May 22 09:44 Beispiel
drwx------  2 wgriege       512 Apr 10 08:37 Noch_eine_Directory
drwx------  2 wgriege       512 Apr 10 08:36 Weitere_Directory
-rw-------  1 wgriege      1675 Apr 10 08:43 ein_file
-rw-------  1 wgriege      5438 Jun 14 22:41 vogel.tif
-rw-------  1 wgriege      7520 Jun 16 22:20 vogel.tif.uu
```

Das File „vogel.tif.uu“ kann nun unbeschadet per **mail** an die Userid „wgriege@gwdg.de“ verschickt werden:

```
mail wgriege@gwdg.de < vogel.tif.uu
```

Die Versendung des kodierten Files „vogel.tif.uu“ über News geschieht mit Hilfe eines Newsreaders, indem das File „vogel.tif.uu“ an einen Artikel angehängt wird.

Die Weiterbehandlung der Files beim Empfänger wird im Kapitel „Weiterbehandlung kopierter oder empfangener Software oder Dokumentation“ erläutert.

8.5.2 Kodierung und Versendung eines Binär-Files in einem Schritt

Selbstverständlich lassen sich die Kodierung und die Versendung eines Binär-Files auch in einem Schritt durchführen. Soll das Binär-File *filename* kodiert an die Userid *userid*@*nodeid* per **mail** verschickt werden, so muß dazu folgendes eingegeben werden:

```
uuencode filename filename |
                    mail userid@nodeid
```

Die Ausgabe des Kommandos **uuencode** wird dabei auf die Standardausgabe geschrieben. Durch das Pipesymbol | wird die Standardausgabe in die Standardeingabe des Kommandos **mail** gelenkt. Das Kommando **mail** schickt dann das kodiert File an die Userid *userid*@*nodeid*.

Für die Versendung über News in einem Schritt kann kein allgemein gültiges Verfahren angegeben werden, da nicht alle Newsreader in einer Pipe benutzbar sind.

8.6 Versendung eines Archiv-Files

Auch mit dem Kommando **tar** erzeugte Archiv-Files und mit **compress** komprimierte Archiv-Files sind Binär-Files in dem Sinne, daß vielfach in den Files das achte Bit eines Zeichens von 0 verschieden ist. Aus diesem Grund können sie nicht ohne eine Kodierung mit **uuencode** per **mail**

oder über das Newssystem versendet werden. Für die Versendung von Archiv-Files sind also die im vorigen Abschnitt erläuterten Schritte einzuhalten. Da Archiv-Files in dieser Einführung allerdings eine besondere Rolle spielen, sind die Schritte im folgenden noch einmal ausführlich dargestellt.

8.6.1 Kodierung eines komprimierten Archiv-Files

Die Kodierung des mit **compress** komprimierten Archiv-Files „Beispiel.tar.Z“ erfolgt durch:

```
uuencode Beispiel.tar.Z
    Beispiel.tar.Z > Beispiel.tar.Z.uu
```

Das kodierte komprimierte Archiv-File befindet sich danach im File „Beispiel.tar.Z.uu“. Der durch das Komprimieren mit **compress** erreichte Effekt der Verkleinerung des Files wird natürlich durch das Kodieren mit **uuencode** wieder geringfügig geschmälert. Trotzdem sollte das Komprimieren nicht unterlassen werden, da insgesamt in der Regel das File „Beispiel.tar.Z.uu“ kleiner ist als das File „Beispiel.tar.uu“, das vor der Kodierung *nicht* komprimiert wurde.

8.6.2 Versendung des kodierten komprimierten Archiv-Files

Die Versendung des kodierten komprimierten Archiv-Files „Beispiel.tar.Z.uu“ per **mail** an die Userid „wgriege@gwdg.de“ geschieht dann durch:

```
mail wgriege@gwdg.de <
                        Beispiel.tar.Z.uu
```

Die Versendung des kodierten Files „Beispiel.tar.Z.uu“ über News geschieht mit Hilfe eines Newsreaders dadurch, daß das File „Beispiel.tar.Z.uu“ an einen Artikel angehängt wird.

Die Weiterbehandlung der Files beim Empfänger wird im Kapitel „Weiterbehandlung kopierter oder empfangener Software oder Dokumentation“ erläutert.

8.6.3 Erzeugung, Komprimierung, Kodierung und Versendung eines Archiv-Files in einem Schritt

Alle vorgestellten Schritte zur Versendung eines Archiv-Files, einschließlich der Erzeugung und Komprimierung, lassen sich zusammenfassen.

Soll das Filesystem *filename* als komprimiertes Archiv-File per **mail** an die Userid *userid@nodeid* geschickt werden, so kann folgende Pipe eingegeben werden:

```
tar -cf - filename | compress |
     uuencode filename.tar.Z |
     mail userid@nodeid
```

Durch die Verwendung des Minuszeichens hinter den Optionen **-cf** des Kommandos **tar**, schreibt **tar** die Ausgabe auf die Standardausgabe.

Durch das Pipesymbol | wird die Standardausgabe in die Standardeingabe des Kommandos **compress** gelenkt. Das Kommando **compress** schreibt die Ausgabe wieder auf die Standardausgabe.

Durch das Pipesymbol | wird die Standardausgabe in die Standardeingabe des Kommandos **uuencode** gelenkt. Da für **uuencode** nur ein Parameter, nämlich *filename***.tar.Z**, angegeben ist, wird von **uuencode** als Filename für das spätere dekodierte File *filename***.tar.Z** eingesetzt. Das Kommando **uuencode** schreibt die Ausgabe wieder auf die Standardausgabe.

Durch das Pipesymbol | wird die Standardausgabe in die Standardeingabe des Kommandos **mail** gelenkt. Das Kommando **mail** verschickt nunmehr das kodierte komprimierte Archiv-File des Filesystems *filename* an die Userid *userid@nodeid*.

Für die Versendung über News in einem Schritt kann kein allgemein gültiges Verfahren angegeben werden, da nicht alle Newsreader in einer Pipe benutzbar sind.

KAPITEL 9 Weiterbehandlung kopierter oder empfangener Software oder Dokumentation

In dem vorliegenden Kapitel wird dargestellt, welche Schritte unternommen werden müssen, damit kopierte oder empfangene Software oder Dokumentation einsatz- oder druckbereit wird.

Beschrieben werden Verfahren zum Kopieren von Software oder Dokumentation von einem Anonymous-FTP-Server sowie zur Behandlung von Software oder Dokumentation, die über Mail oder das Usenet-Newssystem empfangen wurde.

9.1 Von einem Rechner kopierte Software oder Dokumentation

Für das Kopieren von Software oder Dokumentation von einem anderen Rechner stehen häufig Anonymous-FTP-Server zur Verfügung. Deshalb wird das Kopieren von Files von solchen Servern ausführlich beschrieben. Das Kopieren von Files anderer Userids auf fremden Rechnern geschieht analog zum vorgestellten Verfahren.

Im folgenden wird zunächst einmal der Vorgang des Kopierens und danach die Weiterbehandlung der kopierten Software oder Dokumentation beschrieben.

9.1.1 Kopieren von Software oder Dokumentation von einem Anonymous-FTP-Server

Vor dem Kopieren von Software oder Dokumentation von einem Anonymous-FTP-Server sollte zunächst einmal eine für das Kopieren gesonderte Subdirectory eingerichtet werden, um beim späteren Erzeugen eines Filesystems oder von

Files keine Daten unerwünscht zu verlieren, die in gleichnamigen Files enthalten sind.

Beispielsweise sollte demnach die Kommandofolge

```
mkdir Anonymous
cd Anonymous
```

am Bildschirm eingegeben werden. Nun kann mit dem Zugriff auf einen Anonymous-FTP-Server begonnen werden.

Erforderlich ist dabei die Kenntnis der Internet-Adresse desjenigen Rechners, auf dem der ausgewählte Anonymous-FTP-Server installiert ist. Bei der Adresse kann sowohl die IP-Adresse als auch der Internet-Name verwendet werden.

Als Beispiel sei der Anonymous-FTP-Server ausgewählt, der auf einer Workstation des Rechenzentrums der Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG) mit dem Internet-Namen „ftp.gwdg.de“ eingesetzt ist. Die Verbindung zum Anonymous-FTP-Server wird dann mittels des Kommandos **ftp** hergestellt:

```
ftp ftp.gwdg.de
```

Der danach geführte Dialog mit dem Rechner „ftp.gwdg.de“ soll nun anhand des Bildschirmprotokolls dargestellt werden. Als erstes erscheint die Bestätigung der Verbindungsaufnahme:

```
Connected to gwdu03.gwdg.de
220 gwdu03.gwdg.de FTP server (ULTRIX Version 4.1 Tue Mar 19
00:38:17 EST 1991) ready.
Name (ftp.gwdg.de:wgriege): z
```

Der Rechner meldet sich mit seinem eigentlichen Namen, nämlich „gwdu03.gwdg.de“. Der Name „ftp.gwdg.de“ ist nur ein Synonym. Gleichzeitig werden der Name des Betriebssystems, hier Ultrix, und dessen Version bekanntgegeben. Die letzte Zeile fordert zur Eingabe derjenigen Userid auf, unter der die Anmeldung am Rechner „ftp.gwdg.de“

erfolgen soll. Vorgeschlagen wird in diesem Fall die Userid „wgriege“, da auch von einer Userid dieses Namens aus auf den Rechner zugegriffen wird. An dieser Stelle kann eine beliebige auf dem Rechner existierende Userid eingegeben werden. Nach der Eingabe des korrekten Passwords hätte man alle Zugriffsrechte der benutzten Userid.

Da aber hier die Userid „anonymous“ für den Anonymous-FTP-Server verwendet werden soll, wird sie an der Cursor-Position eingetippt:

```
Name (ftp.gwdg.de:wgriege): anonymous
331 Guest login ok, send ident as password.
Password:z
```

In der Regel ist die Eingabe eines Passwords für die Anmeldung bei einem Anonymous-FTP-Server nicht erforderlich. Es gehört jedoch zu den allgemein akzeptierten Sitten, nicht zu verschweigen, wer den Anonymous-FTP-Server in Anspruch nimmt. Aus diesem Grund sollte deshalb an der Cursor-Position die eigene E-Mail-Adresse eingegeben werden, in diesem Beispiel also „wgriege@gwdg.de“. Das Password ist am Bildschirm, im Gegensatz zur hier abgedruckten Darstellung, *nicht sichtbar*:

```
Password:wgriege@gwdg.de
230 Guest login ok, access restrictions apply.
ftp> z
```

Damit ist die Anmeldung beim Anonymous-FTP-Server erfolgt. Es stehen jedoch im Gegensatzt zur Anmeldung unter einer gewöhnlichen Userid nur beschränkte Zugriffsrechte zur Verfügung. Das Kommando **`ftp`** wartet nun auf eine weitere Eingabe und zeigt dies durch den FTP-Prompt „ftp>“ an.

Als nächstes muß in diejenige Subdirectory verzweigt werden, in der die zu kopierende Software aufgeführt ist. Diese zu finden, ist in manchen Fällen nicht einfach, so daß häufiger danach gesucht werden muß. In der Regel steht eine Subdirectory mit dem Namen „pub“ bereit, die öffentlich kopierbare Software oder Dokumentation enthält. Meistens sind darunter weitere Subdirectories vorhanden.

In dem vorgeführten Beispiel soll in die Subdirectory „pub/unix/tools“ verzweigt werden:

```
ftp> cd pub/unix/tools
250 CWD command successful.
ftp> z
```

Nun kann in dieser Working-Directory des Anonymous-FTP-Servers ein Inhaltsverzeichnis durch das FTP-Kommando **`dir`** abgerufen werden:

```
ftp> dir
200 PORT command successful.
150 Opening data connection for /bin/ls (134.76.10.57,2668) (0
bytes).
total 3044
-rw-r--r--  1 292    73    24940 Mar 25  1992 Xtops.tar.Z
-rw-r--r--  1 292    73    26673 Mar 25  1992 a2ps-4.0.tar.Z
-rw-r--r--  1 292    73    48182 Mar 25  1992 arc.tar.Z
-rw-r--r--  1 292    73   323783 Sep 16  1992 f2c-3.2.90.tar.Z
-rw-r--r--  1 292    73    32047 Sep  7  1992 p2c-1.18-1.19.patch.Z
-rw-r--r--  1 292    73   561084 Sep  7  1992 p2c-1.18.tar.Z
-rw-r--r--  1 292    73    28612 Sep  7  1992 p2c-1.19-1.20.patch.Z
-rw-r--r--  1 292    73   631505 Dec 30 09:50 regina-0.03d-gwdg.tar.Z
-rw-r--r--  1 292    73   111825 Mar 25  1992 unzip41.tar.Z
-rw-r--r--  1 292    73   642480 Mar 25  1992 utree_um.sh
-rw-r-----  1 292    73   106045 Mar 25  1992 xbench.tar.Z
-rw-r--r--  1 292    73    44385 Mar 25  1992 xlharc.tar.Z
-rw-r--r--  1 292    73   194823 Mar 25  1992 xloadimage.3.01.tar.Z
-rw-r--r--  1 292    73   244563 Mar 25  1992 zoo210.tar.Z
226 Transfer complete.
1006 bytes received in 0.2 seconds (5 Kbytes/s)
ftp> z
```

Angezeigt werden alle Files, die sich in der Subdirectory „pub/unix/tools“ des Anonymous-FTP-Servers befinden, und das Kommando **ftp** wartet auf eine weitere Eingabe.

Aus dieser Working-Directory der Userid „anonymous“ soll nun das File „Xtops.tar.Z“ in die Working-Directory der Userid „wgriege“ kopiert werden. An der Endung **.tar.Z** im Filenamen ist erkennbar, daß es sich hierbei um ein mit **compress** komprimiertes Archiv-File handelt. Demzufolge ist es ein Binär-File, in dem beim Kopiervorgang das achte Bit eines Zeichens verändert werden könnte. Um eine derartige Veränderung zu verhindern, muß das Kommando **ftp** angewiesen werden, das File „Xtops.tar.Z“ bitweise zu kopieren. Das geschieht dadurch, daß zunächst das FTP-Kommando **binary** eingegeben wird:

```
ftp> binary
200 Type set to I.
ftp> z
```

Mit dem nun einzugebenden FTP-Kommando **get** wird das File „Xtops.tar.Z“ bitweise übertragen:

```
ftp> get Xtops.tar.Z
200 PORT command successful.
150 Opening data connection for Xtops.tar.Z (134.76.10.57,2669)
(24940 bytes).
226 Transfer complete.
local: Xtops.tar.Z remote: Xtops.tar.Z
24940 bytes received in 0.13 seconds (1.9e+02 Kbytes/s)
ftp> z
```

Zusätzlich wird noch die Übertragungsrate angezeigt. In diesem Beispiel beträgt sie 190 KBytes pro Sekunde.

Damit ist die Übertragung abgeschlossen, und der Anonymous-FTP-Server und das Kommando **ftp** können wieder

verlassen werden. Dies geschieht durch die Eingabe des FTP-Kommandos **bye**:

```
ftp> bye
221 Goodbye.
```

Die Verbindung zum Anonymous-FTP-Server ist somit beendet. Das Unix-Kommando

ls -l

liefert nun in der Working-Directory „Anonymous“ die Ausgabe:

```
total 28
-rw-------  1 wgriege     24940 Jun 25 22:06 Xtops.tar.Z
```

Das File „Xtops.tar.Z“ ist demnach erfolgreich vom Anonymous-FTP-Server „ftp.gwdg.de“ kopiert worden.

9.1.2 Informationen über weitere FTP-Kommandos

Alle diejenigen, die ausführliche Informationen über das Unix-Kommando **ftp** erhalten wollen, können durch den Aufruf der zugehörigen Man-Pages

man ftp

eine Beschreibung dieses Kommandos abrufen. Darin sind auch alle FTP-Kommandos beschrieben. Diese kann man auch innerhalb des Programms **ftp** anfordern. Wenn das Programm **ftp** noch nicht aufgerufen ist, gibt man nur

ftp

ein, worauf am Bildschirm der FTP-Prompt erscheint:

```
ftp> z
```

Danach liefert das FTP-Kommando **help** am Bildschirm folgende Ausgabe:

```
ftp> help
Commands may be abbreviated.  Commands are:

!               cr              ls              prompt          runique
$               delete          macdef          proxy           send
account         debug           mdelete         sendport        status
append          dir             mdir            put             struct
ascii           disconnect      mget            pwd             sunique
bell            form            mkdir           quit            tenex
binary          get             mls             quote           trace
bye             glob            mode            recv            type
case            hash            mput            remotehelp      user
cd              help            nmap            rename          verbose
cdup            image           ntrans          reset           ?
close           lcd             open            rmdir
ftp> z
```

Eine Beschreibung zum Beispiel des FTP-Kommandos **dir** erhält man dann durch:

```
ftp> help dir
dir             list contents of remote directory
ftp> z
```

Durch **bye** wird das Kommando **ftp** wieder beendet:

```
ftp> bye
```

9.1.3 Weiterbehandlung der kopierten Software oder Dokumentation

Das File „Xtops.tar.Z“ ist nun erfolgreich vom Anonymous-FTP-Server „ftp.gwdg.de“ kopiert worden.

Aufgrund der Endung **.Z** im Filenamen erkennt man, daß es mit dem Unix-Kommando **compress** komprimiert wurde. Demzufolge muß es zunächst dekomprimiert werden:

```
compress -d Xtops.tar.Z
```

Am Bildschirm erscheinen keine Meldungen, aber das Kommando zum Anzeigen des Inhalts der Working-Directory

```
ls -l
```

liefert die folgende Ausgabe:

```
total 80
-rw-------  1 wgriege     81920 Jun 25 22:06 Xtops.tar
```

Es ist also das dekomprimierte File „Xtops.tar“ entstanden. An der Endung **.tar** dieses Files erkennt man, daß es sich um ein Archiv-File handeln muß, das mit dem Unix-Kommando **tar** erzeugt worden ist. Demzufolge kann es auch mit **tar** wieder in den ursprünglichen Zustand zurückversetzt werden. Dazu gibt man das Kommando

```
tar -xvf Xtops.tar
```

ein. Am Bildschirm erscheinen daraufhin die folgenden Meldungen:

```
x Imakefile, 143 bytes, 1 blocks
x Makefile, 8857 bytes, 18 blocks
x display.h, 1629 bytes, 4 blocks
x XtoPS.c, 40478 bytes, 80 blocks
x encode.c, 12432 bytes, 25 blocks
x XtoPS.man, 2922 bytes, 6 blocks
x README, 800 bytes, 2 blocks
```

Das Kommando

```
ls -l
```

zum Anzeigen des Inhalts der Working-Directory liefert danach die Ausgabe:

```
total 164
-rwx------  1 wgriege       143 Jun 11  1990 Imakefile
-rw-------  1 wgriege      8857 Jun 11  1990 Makefile
-rw-------  1 wgriege       800 Jun 11  1990 README
-rwx------  1 wgriege     40478 Jun 11  1990 XtoPS.c
-rwx------  1 wgriege      2922 Jun 11  1990 XtoPS.man
-rw-------  1 wgriege     81920 Jun 25 22:06 Xtops.tar
-rwx------  1 wgriege      1629 Jun 11  1990 display.h
-rwx------  1 wgriege     12432 Jun 11  1990 encode.c
```

In der Regel stehen dann die weiteren Anweisungen zur Installation der Software in dem File mit dem Namen „README“. Dort sind auch meistens Hinweise zur Anpassung an einzelne Unix-Derivate enthalten.

Nachdem diese Anweisungen befolgt worden sind, kann die Software durch das Unix-Kommando

`make`

einsatzbereit gemacht werden.

Auf analoge Weise wird Dokumentation von einem Anonymous-FTP-Server kopiert. Da sie in der Regel nur aus einem mit **`compress`** komprimierten File besteht, entfällt der Aufruf des Kommandos **`tar`**.

In welchem Format die Dokumentation nach der Dekomprimierung vorliegt, ist meistens an der verbliebenen Endung zu erkennen: Die Endung **`.ps`** bedeutet zum Beispiel, daß das File im PostScript-Format vorliegt und auf einem PostScript-fähigen Drucker ausgedruckt werden kann.

Die Weiterbehandlung der vom Anonymous-FTP-Server kopierten Software oder Dokumentation ist damit abgeschlossen.

9.1.4 Allgemeine Hinweise zur Weiterbehandlung der kopierten Software oder Dokumentation

Wie die Software oder Dokumentation, die von einem Anonymous-FTP-Server kopiert worden ist, weiterbehandelt werden muß, hängt von der Endung des Filenamens der Soft-

ware oder Dokumentation ab. Die folgende Übesicht gibt an, bei welcher Endung welches Programm aufgerufen werden muß:

1. *filename*`.Z` `compress -d` *filename*`.Z`
2. *filename*`.z` `gzip -d` *filename*`.z`
3. *filename*`.gz` `gzip -d` *filename*`.gz`
4. *filename*`.tar` `tar -xvf` *filename*`.tar`
5. *filename*`.shar` `sh` *filename*`.shar`
6. *filename*`.sh` `sh` *filename*`.sh`

Andere Endungen kommen auch vor, sind aber seltener, so daß sie an dieser Stelle nicht aufgeführt werden sollen.

9.2 Empfangene Software oder Dokumentation

Der Empfang von Software oder Dokumentation kann im allgemeinen auf zwei Wegen erfolgen. Zum einen ist es das Mail-System, über das die elektronische Kommunikation geführt wird, womit ein oder mehrere Empfänger gezielt erreicht werden können. Zum anderen ist es das Usenet-Newssystem, auf das viele Empfänger zugreifen können, ohne daß der einzelne Empfänger näher ermittelt werden kann.

In den nächsten Abschnitten soll der Empfang von Software oder Dokumentation aus diesen beiden Quellen näher beschrieben werden.

9.2.1 Empfang eines Shellscript-Archivs

Software oder Dokumentation, die *keine* Binär-Daten enthält, kann als Shellscript-Archiv versendet werden. Diese kann sowohl über Mail als auch über News versendet werden.

Empfang eines Shellscript-Archivs über Mail

Im Kapitel „Versendung kopierbarer Software oder Dokumentation“ wurde gezeigt, wie ein Shellscript-Archiv über Mail versendet wird. Als Beispiel konnte das Shellscript-

Archiv „Beispiel.shar“ verwendet werden, das an die Userid „wgriege“ verschickt wurde. Unter der Userid „wgriege“ kann nun das Unix-Kommando **`mail`** oder ein anderes Mail-Programm, zum Beispiel **`elm`**, benutzt werden, um eine Mail zu empfangen.

Zunächst sollte jedoch eine eigene Subdirectory eingerichtet werden, um beim späteren Erzeugen eines Filesystems oder von Files keine Daten unerwünscht zu verlieren, die in gleichnamigen Files enthalten sind:

```
mkdir Mails
cd Mails
```

Wird das Unix-Kommando **`mail`** verwendet, so genügt zum Empfang zunächst einmal der Aufruf

```
mail
```

Danach erscheint am Bildschirm die folgende Ausgabe:

```
Mail version 2.18 5/19/83.  Type ? for help.
"/usr/spool/mail/wgriege": 1 message 1 new
>N  1 wgriege@gwdg.de Tue Jun 29 21:36  72/1954
& z
```

Das Kommando **`mail`** wartet nun auf eine Eingabe, was mit der eingegangenen Mail geschehen soll, und zeigt dies durch den Prompt „&“ an. Da sie empfangen werden soll, und zwar durch Kopieren in ein File mit dem Namen „mail_file“, wird an der Cursor-Position **`s mail_file`** eingegeben:

```
& s mail_file
"mail_file" [New file] 72/1954
& z
```

Damit ist die eingegangene Mail im File „mail_file“ gespeichert, und das Kommando **mail** kann durch die Eingabe von **q** wieder beendet werden:

```
& q
```

Der Aufruf zum Anzeigen des Inhaltsverzeichnisses der Working-Directory „Mails“

ls -l

zeigt am Bildschirm nun:

```
total 4
-rw-------  1 wgriege      1964 Jun 29 21:36 mail_file
```

Der Inhalt des Files „mail_file“ wird mit

cat mail_file

am Bildschirm angezeigt:

```
From wgriege@gwdg.de Tue Jun 29 21:36:19 1993
Received: by gwdu03.gwdg.de (5.57/1.35gwd1)
        id AA28391; Tue, 29 Jun 93 21:36:17 +0200
Received: by gwdu12.gwdg.de (5.57/Ultrix3.0-C)
        id AA20545; Tue, 29 Jun 93 21:38:32 +0200
Date: Tue, 29 Jun 93 21:38:32 +0200
From: wgriege@gwdg.de (Wilfried Grieger)
Message-Id: <9306291938.AA20545@gwdu12.gwdg.de>
To: wgriege
Status: R

#! /bin/sh
# This is a shell archive, meaning:
# 1. Remove everything above the #! /bin/sh line.
# 2. Save the resulting text in a file.
# 3. Execute the file with /bin/sh (not csh) to create the files:
#       Beispiel
# This archive created: Tue Jun 29 21:38:31 1993
export PATH; PATH=/bin:$PATH
if test ! -d 'Beispiel'
then
        mkdir 'Beispiel'
fi
cd 'Beispiel'
if test ! -d 'Unterdirectory'
then
        mkdir 'Unterdirectory'
fi
cd 'Unterdirectory'
if test -f 'drittes_file'
then
        echo shar: will not over-write existing file "'drittes_file'"
else
cat << \SHAR_EOF > 'drittes_file'
# Dieses ist der Inhalt des ersten Files in der Subdirectory
# "Unterdirectory".
# Der Text ist f"ur das folgende irrelevant.
# Er dient nur zur Illustration.

# Damit endet das File "drittes_file".
SHAR_EOF
fi # end of overwriting check
cd ..
if test -f 'erstes_file'
then
        echo shar: will not over-write existing file "'erstes_file'"
else
cat << \SHAR_EOF > 'erstes_file'
# Dieses ist der Inhalt des ersten Files in der Directory
# "Beispiel".
# Der Text ist f"ur das folgende irrelevant.
# Er dient nur zur Illustration.

# Damit endet das File "erstes_file".
SHAR_EOF
fi # end of overwriting check
if test -f 'zweites_file'
then
        echo shar: will not over-write existing file "'zweites_file'"
else
cat << \SHAR_EOF > 'zweites_file'
```

```
# Dieses ist der Inhalt des zweiten Files in der Directory
# "Beispiel".
# Der Text ist f"ur das folgende irrelevant.
# Er dient nur zur Illustration.

# Damit endet das File "zweites_file".
SHAR_EOF
fi # end of overwriting check
cd ..
#       End of shell archive
exit 0
```

Zum eigentlichen Shellscript-Archiv ist also noch die Information hinzugekommen, die das Mail-System an den Anfang geschrieben hat.

Empfang eines Shellscript-Archivs über News

Ebenso wie über Mail kann ein Shellscript-Archiv über das Usenet-Newssystem versendet werden. Als Beispiel soll wieder das Shellscript-Archiv „Beispiel.shar“ über News empfangen werden. Es ist in die Newsgroup „gwdg.test“ gesendet worden und soll von dort wieder zurückgeholt werden.

Dazu muß ein Newsreader die Newsgroup „gwdg.test“ aufrufen und das Shellscript-Archiv in ein File zum Beispiel mit dem Namen „news_file“ kopieren.

Die Anzeige des Inhaltsverzeichnisses der Working-Directory

```
ls -l
```

liefert nun am Bildschirm die Ausgabe:

```
total 8
-rw-------  1 wgriege      1964 Jun 29 21:36 mail_file
-rw-------  1 wgriege      2076 Jun 30 21:52 news_file
```

Mit dem Kommando

cat news_file

erscheint der Inhalt des neuen Files „news_file“ auf dem Bildschirm:

```
Newsgroups: gwdg.test
Path: gwdu03.gwdg.de!wgriege
From: wgriege@gwdu03.gwdg.de (Wilfried Grieger)
Subject: Shellscript-Archiv Beispiel.shar
Message-ID: <1PXEBUYI@gwdu03.gwdg.de>
Organization: GWDG, Goettingen
Date: Wed, 30 Jun 1993 19:47:24 GMT
Lines: 61

#! /bin/sh
# This is a shell archive, meaning:
# 1. Remove everything above the #! /bin/sh line.
# 2. Save the resulting text in a file.
# 3. Execute the file with /bin/sh (not csh) to create the files:
#       Beispiel
# This archive created: Wed Jun 30 21:47:11 1993
export PATH; PATH=/bin:$PATH
if test ! -d 'Beispiel'
then
        mkdir 'Beispiel'
fi
cd 'Beispiel'
if test ! -d 'Unterdirectory'
then
        mkdir 'Unterdirectory'
fi
cd 'Unterdirectory'
if test -f 'drittes_file'
then
        echo shar: will not over-write existing file "'drittes_file'"
else
cat << \SHAR_EOF > 'drittes_file'
# Dieses ist der Inhalt des ersten Files in der Subdirectory
# "Unterdirectory".
# Der Text ist f"ur das folgende irrelevant.
# Er dient nur zur Illustration.

# Damit endet das File "drittes_file".
SHAR_EOF
fi # end of overwriting check
cd ..
if test -f 'erstes_file'
then
        echo shar: will not over-write existing file "'erstes_file'"
else
cat << \SHAR_EOF > 'erstes_file'
# Dieses ist der Inhalt des ersten Files in der Directory
# "Beispiel".
# Der Text ist f"ur das folgende irrelevant.
# Er dient nur zur Illustration.

# Damit endet das File "erstes_file".
SHAR_EOF
fi # end of overwriting check
if test -f 'zweites_file'
then
        echo shar: will not over-write existing file "'zweites_file'"
```

```
else
cat << \SHAR_EOF > 'zweites_file'
# Dieses ist der Inhalt des zweiten Files in der Directory
# "Beispiel".
# Der Text ist f"ur das folgende irrelevant.
# Er dient nur zur Illustration.

# Damit endet das File "zweites_file".
SHAR_EOF
fi # end of overwriting check
cd ..
#       End of shell archive
exit 0
```

In diesem Fall sind also zum eigentlichen Shellscript-Archiv noch die Informationen des Newssystems hinzugekommen.

In beiden Fällen, beim Empfang über Mail und beim Empfang über News, müssen nun die Kopfzeilen mit Hilfe eines Editors, zum Beispiel mit GNU **emacs**, entfernt werden, und zwar bis einschließlich der Leerzeile vor „#!/bin/sh“.

Weiterbehandlung eines empfangenen Shellscript-Archivs

Erst danach kann das Filesystem wieder extrahiert werden. Das geschieht durch die Eingabe des Kommandos:

sh *filename*

filename ist dabei durch „mail_file“ oder durch „news_file“ zu ersetzen. In beiden Fällen wird das Filesystem „Beispiel“ wiederhergestellt.

Einige Newsreader, beispielsweise der **nn**, besitzen ein Unterkommando, beim **nn** das Kommando **:unshar**, das das Entfernen der Kopfzeilen und das Wiederherstellen des Filesystems nacheinander automatisch erledigt, ohne daß das Shellscript-Archiv erst zwischengespeichert werden muß.

9.2.2 Empfang eines kodierten Binär-Files

Software oder Dokumentation, die Binärdaten enthält, muß vor dem Versenden kodiert werden, damit sie fehlerfrei übertragen wird. Dieses kodierte File kann sowohl über Mail als auch über News versendet und damit auch empfangen werden.

Empfang eines kodierten Binär-Files über Mail

Im Kapitel „Versendung kopierbarer Software oder Dokumentation“ wurde gezeigt, wie ein kodiertes Binär-File über Mail versendet wird. Als Beispiel wurde das Binär-File „vogel.tif“ verwendet, das an die Userid „wgriege“ verschickt wurde.

Unter der Userid „wgriege“ kann nun das Unix-Kommando **`mail`** oder ein anderes Mail-Programm verwendet werden, um eine Mail zu empfangen.

Zunächst sollte jedoch wieder eine eigene Subdirectory eingerichtet werden, um beim späteren Dekodieren des Files keine Daten unerwünscht zu verlieren, die in gleichnamigen Files enthalten sind:

```
mkdir Kodiert
cd Kodiert
```

Wird das Unix-Kommando **`mail`** verwendet, so genügt zum Empfang wieder der Aufruf:

```
mail
```

Danach erscheint am Bildschirm die folgende Ausgabe:

```
Mail version 2.18 5/19/83.  Type ? for help.
"/usr/spool/mail/wgriege": 1 message 1 new
>N  1 wgriege@gwdg.de Thu Jul  1 21:37  135/7879
& z
```

Zum Empfang der Mail durch Kopieren in das File „kod_mail_file“ wird an der Cursor-Position **`s kod_mail_file`** eingegeben:

```
& s kod_mail_file
"kod_mail_file" [New file] 135/7879
& z
```

Damit ist die eingegangene Mail im File „kod_mail_file“ gespeichert, und das Kommando **mail** kann durch die Eingabe von **q** wieder beendet werden:

```
& q
```

Der Aufruf zum Anzeigen des Inhaltsverzeichnisses der Working-Directory „Kodiert“

ls -l

zeigt am Bildschirm nun:

```
total 8
-rw-------  1 wgriege      7889 Jul  1 21:38 kod_mail_file
```

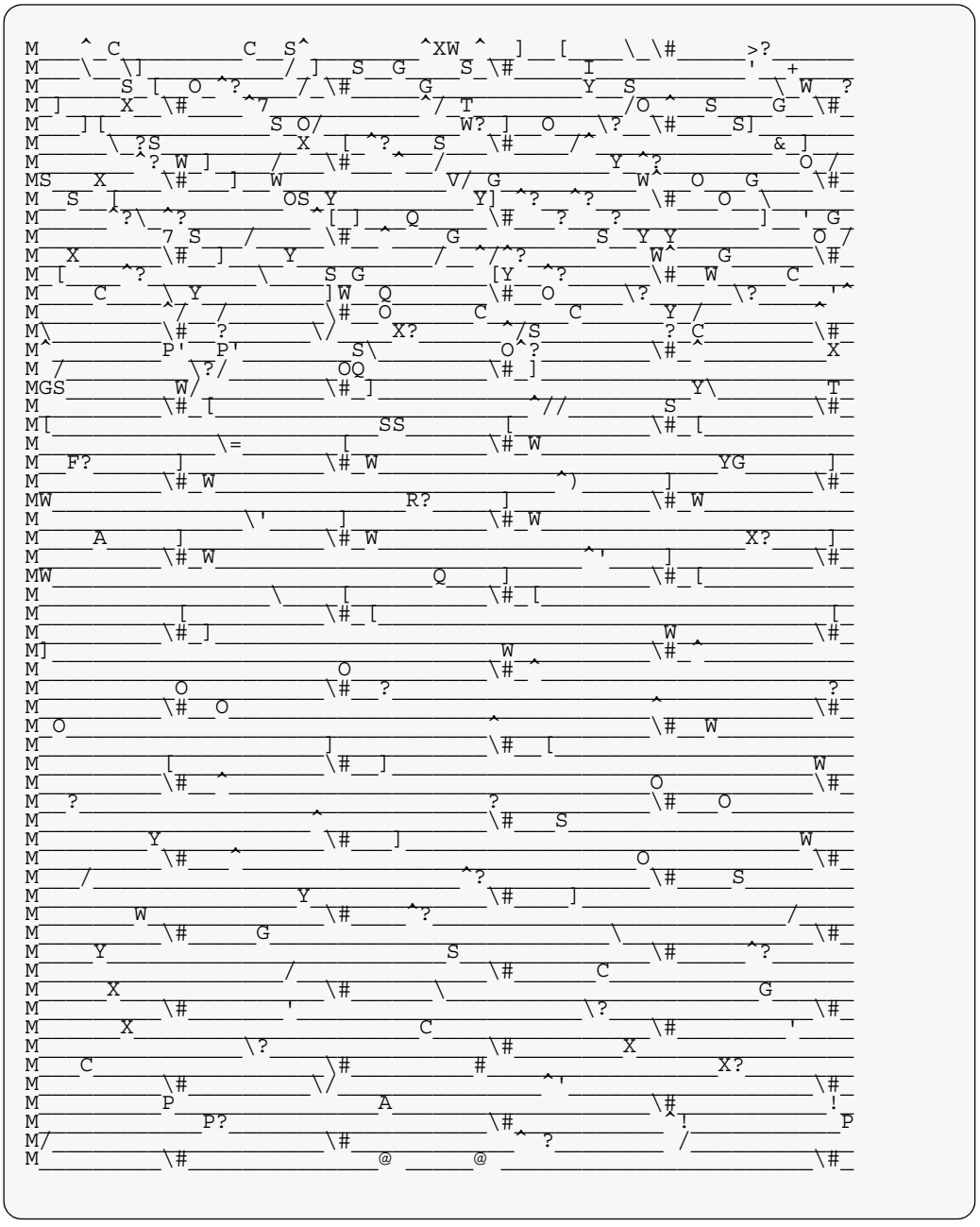
Der Inhalt des Files „kod_mail_file“ wird mit

cat kod_mail_file

am Bildschirm angezeigt:

```
From wgriege@gwdg.de Thu Jul  1 21:37:32 1993
Received: by gwdu03.gwdg.de (5.57/1.35gwd1)
        id AA05615; Thu, 1 Jul 93 21:37:30 +0200
Received: by gwdu12.gwdg.de (5.57/Ultrix3.0-C)
        id AA09178; Thu, 1 Jul 93 21:39:47 +0200
Date: Thu, 1 Jul 93 21:39:47 +0200
From: wgriege@gwdg.de (Wilfried Grieger)
Message-Id: <9307011939.AA09178@gwdu12.gwdg.de>
To: wgriege
Status: R

begin 600 vogel.tif
M24DJ +04  ##%^C__________________________________________\#_
M_____________________________________________\#_____________
M_________________________________\#_________________________
M_____________________\#_____________________________________
M_________\#______________________________________________\#_
M_____________________________________________\#_____________
M_________________________________\#_________________________
M_____________________\#_____________________________________
M_________\#_________________________________^____________\#_
M________________________________^____________\#_____________
M____________________]____________\#_________________________
M________]____________\#_________________________________[___
M_________\#_________________________________[____________\#_
M__________X  /__________________W____________\#__________P'_
M_P'_________________W____________\#_________\/____X?________
M________O____________\#_________C______C________________O__W
M_________\#________\?______\?_______________?  O_________\#_
M_______C________C_______________?  O_________\#_______^?___C
M____\______________^___?_________\#_______Y____=_____/______
M_______^__^__________\#_______G___^^_____S_____________]__]_
M_________\#_______?___^^_____]_____________]__]___W______\#_
M_____\____^^_____^?_____________[__[__O______\#______[_____=
M______O____________[__W___?______\#______W_____C______W_____
M_______[__W__^_______\#______/____________Y____________W__O_
M_]_______\#_____^_____________^____________W__?__[_______\#_
M____]______________?___________O_^___W___W___\#_____[_______
M_______O___________O_^___W___O___\#_____W______________W____
M_______?_]___O___?___\#_____W_\?___________W___________?_[__
M_?__\____\#_____O_[O___________[__________^__[___^__[____\#_
M____?_WW___________]__________^__W__]___W____\#____^__WW____
M_______^__________]__O___[__O____\#_____^_WW___________^____
M______]__?__W___?____\#____]__[O____________?_________[__?__
MO__^_____\#____]__\?___^________?_________[_^___?__]_____\#_
M___]________/_______?_________W_]__^___S_____\#____[________
MS_______O_________W_]__]___O_____\#_____[_______\_______O___
M______O_[___[__?_____\#____[_________/______O_________O_W__W
M__^______\#____W_________S______W_________O_O__O__]______\#_
M___W_____S___\______W_________?_O__O__[___W__\#____W_____\__
M__/_____W_________?_?__?__W___/__\#____W______'___S_____7___
M_____^_^__^___/__^___\#____W_______Y_@______@________^_^__]_
M_^___Y___\#_____W_____^>_?^?____W__/_____]_]__[__]___W___\#_
M___W______^!___G____W__P/____]__[__W__[__/___\#____W______!\
M___Y____W___P?____[W__O__W__\____\#____W_____P___'_^?___W___
M_A___[_W__?__O___[___\#____[____\/___Y___G__O_____^'_W_O_^__
M_?__G____\#_____[___#___^?__Y___O_____X?_W_?_]___\__?____\#_
M____[__X_____C__^?__O______C_O_?_[___[_\_____\#____]___'____
M_\___G__?______\?O^__W__W__[_____\#____]__X_______'__Y__?___
M____C?]___O_O__G_____\#____]__'_______Y___^_?_______\?[__?__
M?__?_____\#____^_\________^?__/^________^G[__?_^___\_____\#_
```



```
M_________________     ?_____________________________\#_____________
M_____________________________\#_____________________________________
M_____________________\#_____________________________________________
M__________\#_____________________________________________\#_
M_____________________________________\#_____________
M_____________________________\#_____________________
M_____________________\#_____________________________
M__________\#_____________________________________\#_
M_____________________________________\ +   ! P !
M&@$   $! P !    DP    (! P !     0    ,! P !     0    8! P !
M     0   !$!!  !    "    !(! P !     0   !4! P !     0   !8!
F!  !    DP   !<!!  !    K!0  !P! P !     0

end
```

Diese Bildschirmausgabe ist zwar wenig informativ, sollte der Vollständigkeit halber jedoch einmal dargestellt werden. Am Anfang des Files steht also die Information, die das Mail-System hinzugefügt hat.

Empfang eines kodierten Binär-Files über News

Ebenso wie über Mail kann ein kodiertes Binär-File auch analog über das Usenet-Newssystem versendet und kodierten Als Beispiel soll wieder das kodierte File „vogel.jpg“ aus den News empfangen werden. Es ist in die Newsgroup „gwdg.test“ gesendet worden und soll von dort wieder zurückgeholt werden. Dazu muß ein Newsreader die Newsgroup „gwdg.test“ aufrufen und das kodierte File in ein File zum Beispiel mit dem Namen „kod_news_file“ kopieren.

Die Anzeige des Inhaltsverzeichnisses der Working-Directory

```
ls -l
```

liefert dann am Bildschirm die Ausgabe:

```
total 16
-rw-------  1 wgriege      7889 Jul  1 21:38 kod_mail_file
-rw-------  1 wgriege      7999 Jul  1 22:41 kod_news_file
```

Mit dem Kommando

```
cat kod_news_file
```

erscheint der Inhalt des neuen Files „kod_news_file“ auf dem Bildschirm. Diesmal sind aufgrund der großen Länge des Files nur der Anfang und das Ende des Files dargestellt:

```
Newsgroups: gwdg.test
Path: gwdu03.gwdg.de!wgriege
From: wgriege@gwdu03.gwdg.de (Wilfried Grieger)
Subject: Kodiertes File "vogel.tif"
Message-ID: <CVYEBBBQ@gwdu03.gwdg.de>
Organization: GWDG, Goettingen
Date: Thu, 1 Jul 1993 20:35:22 GMT
Lines: 124

begin 600 vogel.tif
M24DJ +04  ##%^C_____________________________________\#_
M_____________________________________\#_____________
M_____________________\#_____________________________
M_________\#_____________________________________\#_
M_____________________________________\#_____________
M_____________________\#_____________________________
M_____________________\#_____________^_____________\#_
M_____________________^_____________\#_____________
M_____________________]_____________\#_____________
M_____________]_____________\#_____________[_____
M_____________\#_____________[_____________\#_
M_____________X  /_____________W_____________\#_____P'_
M_P'_____________W_____________\#_____\/____X?_____
M_____O_____________\#_____C_____C_____O_____W
M_____\#_____\?_____\?_____?__O_____\#_____^?___C
M_____C_____C_____?__O_____\#_____^?___C
.
.
.
.
.
M_____P?_____\#_____^!_____P
M/_____\#_____^_?_____/_____
M_____\#_____@_____@_____\#_
M_____?_____\#_____
M_____\#_____
M_____\#_____
M_____\#_____
M_____\#_____
M_____\#_____
M_____\#_____
M_____\#_____
M_____\ +   ! P !
M&@$   $! P !   DP    (! P !    0    ,! P !    0    8! P !
M     0   !$!!  !    "    !(! P !     0   !4! P !     0   !8!
F!  !    DP   !<!!  !    K!0  !P! P !     0

end
```

In diesem Fall sind also zum eigentlichen kodierten File noch die Informationen des Newssystems hinzugekommen.

In beiden Fällen, beim Empfang über Mail oder beim Empfang über News, brauchen die zusätzlichen Informationen in den Kopfzeilen der Files *nicht* entfernt zu werden.

Weiterbehandlung eines empfangenen kodierten Binär-Files

Es genügt einfach der Aufruf

uudecode *filename*

zum Dekodieren der Files. *filename* ist dabei durch „kod_mail_file“ oder durch „kod_news_file“ zu ersetzen. Erzeugt wird danach das File „vogel.tif“, was man durch

ls -l

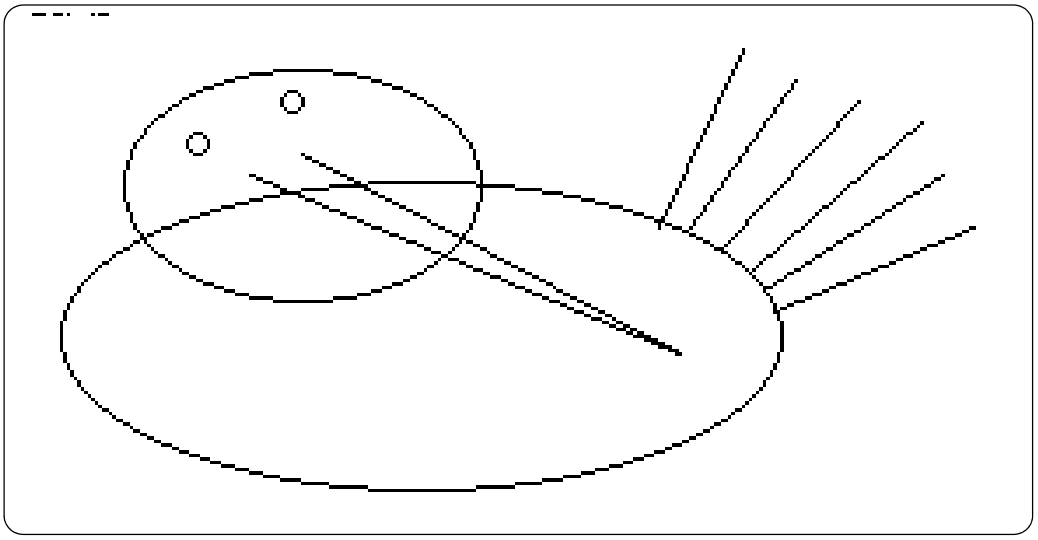
im Inhaltsverzeichnis der Working-Directory erkennen kann:

```
total 24
-rw-------  1 wgriege      7889 Jul  1 21:38 kod_mail_file
-rw-------  1 wgriege      7999 Jul  1 22:41 kod_news_file
-rw-------  1 wgriege      5438 Jul  1 23:23 vogel.tif
```

Der Aufruf des Programms

xv vogel.tif

liefert dann am Bildschirm wieder die Ausgabe:



Einige Newsreader, beispielsweise der **nn**, besitzen ein Unterkommando, beim **nn** das Kommando **:decode**, das das Empfangen und Dekodieren nacheinander automatisch erledigt, ohne daß das kodierte File erst zwischengespeichert werden muß.

Häufig werden Binär-Files auch in mehreren Teilen versendet. In diesem Fall müssen alle Teile in der richtigen Reihenfolge hintereinander in das gleiche File *filename* kopiert werden. Die Informationen, die das Mail- oder das Newssystem einfügen und die sich nun zwischen den Teilen befinden, müssen mit Hilfe eines Editors, zum Beispiel mit dem GNU **emacs**, gelöscht werden. Danach kann mit

uudecode *filename*

das File *filename* dekodiert werden.

Bei einigen Newsreadern, zum Beispiel beim **nn**, wird auch diese Aufgabe durch ein Unterkommando, beim **nn** ebenfalls durch **:decode**, in einem Schritt erledigt.

Auf diese Weise können selbstverständlich auch kodierte komprimierte Archiv-Files empfangen werden. Nach der Dekodierung muß dann in der Regel anhand der erzeugten Endung des Filenamens entschieden werden, ob und mit welchen Kommandos oder Utilities das File weiterbehandelt werden kann.

ANHANG A `tar` zum Nachschlagen

Im folgenden soll zusammenfassend die allgemeine Syntax des Aufrufs des Kommandos **`tar`** zum Archivieren von Files oder Filesystemen in ein Archiv-File dargestellt werden. Für das Kommando **`tar`** sind neben den im Kapitel „**`tar`** - zur Archivierung von Files und Filesystemen“ vorgestellten Optionen noch weitere wählbar.

Eine Übersicht über alle wesentlichen Optionen, die unter allen Unix-Derivaten verfügbar sind, wird mit Erläuterungen aufgestellt. Eigenschaften, die eine Magnetbandverarbeitung betreffen, sind nicht aufgeführt.

A.1 Grundoptionen des Kommandos `tar`

Das Kommando **`tar`** besitzt die in der folgenden Tabelle aufgeführten fünf Grundoptionen. In manchen Unix-Derivaten werden sie auch als *function keys*, *function letters* oder als *required flags* bezeichnet.

Mit einer Grundoption wird eine Funktion des Kommandos **`tar`** ausgewählt. Deshalb muß in jedem Fall genau eine dieser Grundoptionen, und zwar als erste Option, angegeben werden.

TABELLE 1 Grundoptionen des Kommandos `tar`

Grundoption	*Erläuterung*
`-c`	Erzeugung („create“) eines neuen Archivs. Falls ein Archiv gleichen Namens bereits besteht, wird es ohne Warnung überschrieben.
`-r`	Erweiterung („write“) eines bereits bestehenden Archivs mit neuen Files.
`-t`	Anzeige („type“) aller Files in einem Archiv, und zwar in der Reihenfolge, in der sie archiviert wurden. Mehrfach archivierte Files werden dabei auch mehrfach angezeigt.
`-u`	Austausch („update“) von Files in einem bereits bestehenden Archiv durch Files gleichen Namens, aber mit neuerem Änderungsdatum. Ist das Änderungsdatum des zu archivierenden Files älter als das des im Archiv befindlichen Files, so findet kein Austausch statt. Ist der Name des auszutauschenden Files im Archiv nicht vorhanden, so wirkt die Option **`-u`** wie die Option **`-r`**.

TABELLE 1 Grundoptionen des Kommandos `tar`

Grundoption	*Erläuterung*
`-x`	Extraktion („extract“) von Files aus einem Archiv. Benötigte Directories werden rekursiv in der Working-Directory angelegt. Angaben zum Eigentümer, zum letzten Änderungsdatum und zu den Zugriffsrechten werden soweit wie möglich wiederhergestellt. Wird kein File angegeben, das extrahiert werden soll, so wird das gesamte Archiv wiederhergestellt. Falls mehrere Files gleichen Namens im Archiv vorhanden sind und nicht zusätzlich die Option **`-w`** verwendet wird, so überschreibt das zuletzt archivierte alle vorhergehenden Versionen.

A.2 Allgemeine Syntax

Das Unix-Kommando **`tar`** besitzt die oben erläuterten fünf Grundoptionen, von denen genau eine, und zwar als erste Option, angegebenen werden muß. Durch die Angabe einer Grundoption werden die in den folgenden Abschnitten aufgeführten Funktionen des Kommandos **`tar`** ausgewählt.

A.2.1 Zur Erzeugung eines Archivs

Soll das Filesystem *`filename`* erstmalig in das Archiv-File *`filename`***`.tar`** archiviert werden, so lautet dafür der Aufruf des Kommandos:

`tar -c`*`opt`***`f`** *`filename`***`.tar`** *`filename`*

Anstelle von *`opt`* sind weitere Optionen einsetzbar. Es ist auch möglich, einzelne Files oder einzelne Files zusammen mit ganzen Filesystemen in einem Archiv-File zu archivieren. In diesem Fall muß der Parameter *`filename`* durch die

Namen der zu archivierenden Files oder Filesysteme ersetzt werden.

A.2.2 Zur Erweiterung eines Archivs

Soll das Archiv-File *filename*.**tar** durch die Files oder Filesysteme *file1*, *file2*, ..., *filen* erweitert werden, so lautet dafür der Aufruf des Kommandos:

```
tar -ropt f filename.tar file1 file2
                                  ... filen
```

Anstelle von *opt* sind weitere Optionen einsetzbar.

A.2.3 Zum Anzeigen des Inhalts eines Archivs

Soll der Inhalt eines Archiv-Files *filename*.**tar** angezeigt werden, so lautet dafür der Aufruf des Kommandos:

```
tar -topt f filename.tar
```

Anstelle von *opt* sind weitere Optionen einsetzbar.

A.2.4 Zum Austauschen von Files in einem Archiv

Sollen die Files oder Filesysteme *file1*, *file2*, ..., *filen* in einem bereits bestehenden Archiv-File *filename*.**tar** durch neuere Versionen ersetzt werden, so lautet dafür der Aufruf des Kommandos:

```
tar -uopt f filename.tar file1 file2
                                  ... filen
```

Anstelle von *opt* sind weitere Optionen einsetzbar.

A.2.5 Zur Extraktion von Files aus einem Archiv

Sollen die Files oder Filesysteme *file1*, *file2*, ..., *filen* aus einem Archiv-File *filename*.**tar** extrahiert werden, so lautet dafür der Aufruf des Kommandos:

```
tar -xopt f filename.tar file1 file2
                                  ... filen
```

Wird *file1 file2 ... filen* weggelassen, so wird das gesamte Archiv-File *filename*.**tar** extrahiert.

Anstelle von *opt* sind weitere Optionen einsetzbar.

A.3 Weitere Optionen des Kommandos `tar`

Die folgende Tabelle enthält diejenigen Optionen des Kommandos **`tar`**, die unter den meisten Unix-Derivaten zur Verfügung stehen. Optionen, die eine Magnetbandverarbeitung betreffen, sind nicht aufgeführt. Es wird vorausgesetzt, daß die Userid, die das Kommando **`tar`** ausführt, keine Superuser-Privilegien besitzt.

TABELLE 2 Weitere Optionen des Kommandos `tar`

Option	*Erläuterung*
`-f`	Der unmittelbar danach folgende erste Parameter ist der Name des Archiv-Files und kein Archiv auf einem Magnetband. Wenn als Name des Archiv-Files ein Minuszeichen **`-`** gewählt wird, so liest das Kommando von der Standardeingabe bzw. schreibt auf die Standardausgabe; je nach angegebener Grundoption. Damit kann das Kommando auch als Filter verwendet werden.
`-l`	Meldung aller nicht auflösbaren Links. Wird die Option nicht angegeben, werden auch keine diesbezüglichen Fehlermeldungen angezeigt.
`-m`	Die im Archiv gespeicherten Änderungsdaten der Files werden nicht restauriert. Als Änderungsdatum wird das Extraktionsdatum angegeben.

TABELLE 2 Weitere Optionen des Kommandos `tar`

Option	*Erläuterung*
`-p`	Die zu extrahierenden Files werden mit den im Archiv gespeicherten Zugriffrechten restauriert. Die aktuelle **`umask`**-Einstellung wird ignoriert.
`-v`	Meldung aller Aktionen des Kommandos.
`-w`	Meldung aller Aktionen des Kommandos. Nach jeder Aktion wird auf eine Bestätigung gewartet. Beginnt die Bestätigung mit dem Buchstaben „y“, so wird die Aktion durchgeführt, sonst nicht.

A.4 Man-Pages des Kommandos `tar`

Üblicherweise findet man unter den Unix-Derivaten die Erläuterungen zu einzelnen Kommandos in den Man-Pages.

Um die genaue Beschreibung des Kommandos **`tar`** am Bildschirm lesen zu können, muß folgendes Kommando eingegeben werden:

```
man tar
```

Danach erscheint die ausführliche Beschreibung am Bildschirm. Mit den üblichen Umlenkmechanismen, zum Beispiel mit

```
man tar > tar.man
```

läßt sich die Ausgabe in ein File mit dem in diesem Beispiel gewählten Namen „tar.man“ umlenken. Danach kann der Inhalt des Files auch gedruckt werden.

Die Erläuterungen zum Aufbau eines Archiv-Files findet man in der Regel in der fünften Sektion der Man-Pages, so daß man mit

```
man 5 tar
```

eine diesbezügliche Beschreibung erhält. Unter dem Betriebssystem OSF/1 ist dafür dagegen die vierte Sektion vorgesehen:

```
man 4 tar
```

Unter SCO-Unix wird dafür die Sektion F verwendet:

```
man F tar
```

Unter anderen Unix-Derivaten, zum Beispiel AIX, fehlt diese Beschreibung vollständig.

A.5 Besonderheiten unter einigen Unix-Derivaten

Zusätzlich zu den im vorigen Abschnitt erläuterten Besonderheiten in den Man-Pages der verschiedenen Unix-Derivate gibt es auch Besonderheiten bezüglich des Umfangs der wählbaren Optionen.

Diese zusätzlichen Möglichkeiten zur Tabelle „Weitere Optionen des Kommandos **tar**“ sind im folgenden dargestellt. Besonderheiten bei einer Magnetbandverarbeitung sind nicht aufgeführt.

A.5.1 IBM AIX

Unter dem Betriebssystem AIX stehen die folgenden weiteren Optionen des Kommandos **tar** zusätzlich zu den in der

Tabelle „Weitere Optionen des Kommandos **`tar`**“ erwähnten zur Verfügung.

TABELLE 3 Weitere Optionen des Kommandos `tar` unter AIX

Option	*Erläuterung*
`-C`	Vor der Archivierung der angegebenen Files wird zunächst in die unmittelbar hinter dieser Option angegebene Directory verzweigt. Das hat den Vorteil, daß nur kurze relative Pfadnamen im Archiv verzeichnet werden.
`-F`	Alle sogenannten SCCS-Directories, core-Files und error-Files werden ignoriert.
`-L`	Die unmittelbar hinter dieser Option angegebenen Files werden hinter den sonst angegebenen Files archiviert. Directories werden zwar archiviert, aber nicht, wie sonst üblich, rekursiv mit allen Subdirectories und Files.
`-d`	Auch sogenannte Special-Files werden archiviert.
`-h`	Anstelle des Namens des symbolischen Link-Files wird eine vollständige Kopie des Link-Files archiviert.
`-i`	Im Archiv gefundene Prüfsummenfehler werden ignoriert.
`-s`	Falls ein regulärer Link zwischen zwei Files nicht möglich ist, wird versucht, einen symbolischen Link zwischen den beiden Files aufzubauen.

A.5.2 Linux

Unter dem Betriebssystem Linux wird für das Kommando **tar** in der Regel das von der Free Software Foundation entwickelte verbesserte Kommando GNU **gtar** verwendet. Aus diesem Grund entsprechen die Optionen des Kommandos **tar** unter Linux denjenigen des Kommandos **gtar**.

A.5.3 SCO-Unix

SCO-Unix unterscheidet sich in zwei Eigenschaften von den im Kapitel „**tar** - zur Archivierung von Files und Filesystemen“ erläuterten:

1. Im Archiv-File können auch absolute Filenamen registriert werden. Das läßt sich durch die Wahl der Option **-A** vermeiden.
2. Wird beim Extrahieren die Option **-q** gewählt, so kann damit auch die älteste Version eines Files wiederhergestellt werden.

Die folgenden Optionen des Kommandos **tar** stehen zusätzlich zu den in der Tabelle „Weitere Optionen des Kommandos **tar**“ erwähnten zur Verfügung.

TABELLE 4 Weitere Optionen des Kommandos tar unter SCO-Unix

Option	*Erläuterung*
-A	Bei Filenamen werden führende „/“ entfernt.
-F	Das unmittelbar hinter dieser Option angegebene File enthält weitere Argumente für das Kommando.
-L	Anstelle des Namens des symbolischen Link-Files wird eine vollständige Kopie des Link-Files archiviert.

TABELLE 4 Weitere Optionen des Kommandos `tar` unter SCO-Unix

Option	*Erläuterung*
`-q`	Die Extraktion eines Files endet unmittelbar nach der Extraktion des zuerst gefundenen Files, ohne daß nach weiteren Files mit dem gleichen Namen gesucht wird. Damit wird automatisch nur die älteste Version eines Files wiederhergestellt.

A.5.4 SunOS

Unter dem Betriebssystem SunOS stehen die folgenden weiteren Optionen des Kommandos **`tar`** zusätzlich zu den in der Tabelle „Weitere Optionen des Kommandos **`tar`**“ erwähnten zur Verfügung.

TABELLE 5 Weitere Optionen des Kommandos `tar` unter SunOS

Option	*Erläuterung*
`-F`	Alle sogenannten SCCS-Directories, core-Files und error-Files werden ignoriert.
`-FF`	Zusätzlich zu allen SCCS-Directories, core-Files und error-Files werden alle Files mit dem Namen „a.out“ und alle Files, die auf **`.o`** enden, ignoriert.
`-X`	Das unmittelbar hinter dieser Option angegebene File enthält eine Liste von Files oder Directories, die bei der Verwendung der Grundoptionen **`-c`**, **`-t`** oder **`-x`** ignoriert werden sollen.

TABELLE 5 Weitere Optionen des Kommandos `tar` unter SunOS

Option	*Erläuterung*
`-e`	Falls unerwartete Fehler auftauchen, wird das Kommando mit einem positiven Exit-Status abgebrochen.
`-h`	Anstelle des Namens des symbolischen Link-Files wird eine vollständige Kopie des Link-Files archiviert.
`-i`	Im Archiv gefundene Prüfsummenfehler werden ignoriert.
`-o`	Informationen über den Eigentümer und den Gruppennamen aus dem Archiv werden unterdrückt.

A.5.5 DEC Ultrix

Unter dem Betriebssystem Ultrix stehen die folgenden weiteren Optionen des Kommandos **`tar`** zusätzlich zu den in der Tabelle „Weitere Optionen des Kommandos **`tar`**“ erwähnten zur Verfügung.

TABELLE 6 Weitere Optionen des Kommandos `tar` unter Ultrix

Option	*Erläuterung*
`-A`	Der unmittelbar danach angegebene Parameter ist die Nummer des Files im Archiv, mit dem die Ausgabe beginnen soll.

TABELLE 6 Weitere Optionen des Kommandos `tar` unter Ultrix

Option	*Erläuterung*
`-C`	Vor der Archivierung der angegebenen Files wird zunächst in die unmittelbar hinter dieser Option angegebene Directory verzweigt. Das hat den Vorteil, daß nur kurze relative Pfadnamen im Archiv verzeichnet werden.
`-D`	Directories werden in einem früher gebräuchlichen tar-Format ausgegeben.
`-F`	Alle sogenannten SCCS-Directories, core-Files und error-Files werden ignoriert.
`-FF`	Zusätzlich zu allen SCCS-Directories, core-Files und error-Files werden alle Files mit dem Namen „a.out“ und alle Files, die auf **`.o`** enden, ignoriert.
`-H`	Anzeige einer Zusammenfassung aller Grundoptionen und aller weiteren Optionen („Help“).
`-M`	Der unmittelbar danach angegebene Parameter ist die maximale Zahl der Files, die in das Archiv geschrieben werden dürfen. Die aktuelle Zahl der Files im Archiv wird ausgegeben.

TABELLE 6 Weitere Optionen des Kommandos `tar` unter Ultrix

Option	*Erläuterung*
-O	Eigentümer und Gruppennamen werden bei der Wahl der Grundoptionen **-t** oder **-x** mit angezeigt. Eine Warnung wird ausgegeben, wenn der Eigentümer nicht in „/etc/passwd“ oder der Gruppenname nicht in „/etc/group“ gefunden wird.
-R	Der unmittelbar danach angegebene Parameter ist der Filename eines Files, das eine Liste von Files enthält, die bei der Wahl der Grundoptionen **-c** oder **-x** nur berücksichtigt werden sollen.
-V	Ausgabe ausführlicher Informationen, die auch die Informationen, die nach der Angabe der Option **-v** ausgegeben werden, umfassen.
-h	Anstelle des Namens des symbolischen Link-Files wird eine vollständige Kopie des Link-Files archiviert.
-i	Im Archiv gefundene Prüfsummenfehler werden ignoriert.
-o	Informationen über den Eigentümer und den Gruppennamen aus dem Archiv werden unterdrückt.

A.5.6 OSF/1

Unter dem Betriebssystem OSF/1 stehen die folgenden weiteren Optionen des Kommandos **tar** zusätzlich zu den in

der Tabelle „Weitere Optionen des Kommandos **tar**“ erwähnten zur Verfügung.

TABELLE 7 Weitere Optionen des Kommandos `tar` unter OSF/1

Option	*Erläuterung*
-C	Vor der Archivierung der angegebenen Files wird zunächst in die unmittelbar hinter dieser Option angegebene Directory verzweigt. Das hat den Vorteil, daß nur kurze relative Pfadnamen im Archiv verzeichnet werden.
-F	Alle sogenannten SCCS-Directories, alle RCS-Files, alle Files mit den Namen „core“, „errs“, „a.out“ und alle Files, die auf **.o** enden, werden ignoriert.
-L	Falls ein regulärer Link zwischen zwei Files nicht möglich ist, wird versucht, einen symbolischen Link zwischen den beiden Files aufzubauen.
-e	Der unmittelbar danach angegebene Parameter ist der Name des Files, das nicht archiviert werden soll.
-h	Anstelle des Namens des symbolischen Link-Files wird eine vollständige Kopie des Link-Files archiviert.
-i	Im Archiv gefundene Prüfsummenfehler werden ignoriert.
-o	Informationen über den Eigentümer und den Gruppennamen aus dem Archiv werden unterdrückt.

A.6 Grundoptionen des Kommandos `gtar`

Das von der Free Software Foundation entwickelte Kommando zur Archivierung von Files oder Filesystemen GNU **`gtar`** besitzt die in der folgenden Tabelle aufgeführten Grundoptionen.

Dabei sind auch Langformen in der Schreibweise der Optionen angebbar, die leichter zu merken sind als die üblichen kurzen Formen.

TABELLE 8 Grundoptionen des Kommandos `gtar`

Grundoption	*Erläuterung*
`-c` **`+create`**	Erzeugung („create“) eines neuen Archivs. Falls ein Archiv gleichen Namens bereits besteht, wird es ohne Warnung überschrieben.
`-d` **`+compare`** **`+diff`**	Der Inhalt des Archiv-Files wird mit dem angegebenen Filesystem verglichen. Differenzen werden angezeigt.
`-r` **`+append`**	Erweiterung („write“) eines bereits bestehenden Archivs mit neuen Files.
`-t` **`+list`**	Anzeige („type“) aller Files in einem Archiv, und zwar in der Reihenfolge, in der sie archiviert wurden. Mehrfach archivierte Files werden dabei auch mehrfach angezeigt.

TABELLE 8 Grundoptionen des Kommandos `gtar`

Grundoption	*Erläuterung*
`-u` **`+update`**	Austausch („update“) von Files in einem bereits bestehenden Archiv durch Files gleichen Namens, aber mit neuerem Änderungsdatum. Ist das Änderungsdatum des zu archivierenden Files älter als das des im Archiv befindlichen Files, so findet kein Austausch statt. Ist der Name des auszutauschenden Files im Archiv nicht vorhanden, so wirkt die Option **`-u`** wie die Option **`-r`**.
`-x` **`+extract`** **`+get`**	Extraktion („extract“) von Files aus einem Archiv. Benötigte Directories werden rekursiv in der Working-Directory angelegt. Der Eigentümer, das letzte Änderungsdatum und die Zugriffsrechte werden soweit wie möglich wiederhergestellt. Wird kein File angegeben, das extrahiert werden soll, so wird das gesamte Archiv wiederhergestellt. Falls mehrere Files gleichen Namens im Archiv vorhanden sind und nicht zusätzlich die Option **`-w`** verwendet wird, so überschreibt das zuletzt archivierte alle vorhergehenden Versionen.
`-A` **`+catenate`** **`+concatenate`**	Mehrere Archiv-Files werden zu einem großen Archiv-File verbunden.
`-D` **`+delete`**	Das angegebene File wird im Archiv gelöscht.

Die weiteren Optionen des Kommandos **gtar** entnehme man der mit der Software ausgelieferten Beschreibung.

ANHANG B `shar` zum Nachschlagen

Im folgenden soll zusammenfassend die allgemeine Syntax des Aufrufs der Utility **`shar`** zum Archivieren von Files oder Filesystemen in einem Shellscript-Archiv dargestellt werden. Ebenso ist die allgemeine Syntax zum Wiederherstellen der Files oder der File-systeme aus einem Shellscript-Archiv zusammengefaßt.

Für die Utility **`shar`** sind noch andere Optionen als die im Kapitel „**`shar`** - zur Zusammenfassung von Files in einem Shellscript“ vorgestellte Option **`-a`** wählbar. Deshalb sind auch alle verfügbaren Optionen erläutert.

B.1 Allgemeine Syntax

Im folgenden ist die allgemeine Syntax der Utility **`shar`** zum Archivieren von Files oder Filesystemen in einem Shellscript-Archiv dargestellt. Die Wiederherstellung der Files oder der Filesysteme geschieht mit dem Unix-Kommando **`sh`**.

B.1.1 Allgemeine Syntax der Utility `shar`

Ein Filesystem *`filename`* und weitere Files oder Filesysteme *`file1`*, *`file2`*, *`...`*, *`filen`* werden durch folgenden Aufruf der Utility **`shar`** in ein Shellscript-Archiv mit dem Namen *`filename`***`.shar`** archiviert:

```
shar -optionen filename file1 file2
            ... filen > filename.shar
```

Die wählbaren Optionen **`-`***`optionen`* entnehme man dem nächsten Abschnitt.

B.1.2 Allgemeine Syntax zum Wiederherstellen aus einem Shellscript-Archiv

Ein Shellscript-Archiv mit dem Namen *filename*.**shar** wird durch folgenden Kommandoaufruf wieder in seine ursprüngliche Form zerlegt:

```
sh filename.shar
```

Als Alternative kann eine Bourne-Shell als Subshell eröffnet werden:

```
sh
chmod u+x filename.shar
filename.shar
C-d
```

Durch das letzte Kommando **C-d** (Control-d) wird die Bourne-Shell nach der Extraktion wieder verlassen.

Die Zahl der erscheinenden Meldungen richtet sich nach den gewählten Optionen, die beim Archivieren der Utility **shar** mitgegeben wurden.

B.2 Optionen der Utility **shar**

Die Utility **shar** besitzt die in der folgenden Tabelle angegebenen Optionen.

TABELLE 9 Optionen der Utility shar

Option	*Erläuterung*
-a	Enspricht den Optionen **-bcv -p <Tab>X**, wobei **<Tab>** für das Tabulatorzeichen steht.

TABELLE 9 Optionen der Utility `shar`

Option	*Erläuterung*
`-b`	Sind beim Archivieren absolute Pfadnamen angegeben, so wird im Shellscript-Archiv die Root-Directory „/“ weggelassen. Wenn danach ganze Directories extrahiert werden, kann das zu unerwünschten Nebeneffektien führen.
`-c`	Vermerk der Filegrößen durch Auszählen der Zeichen. Stimmt die Größe der extrahierten Files nicht mit den eingetragenen Größen überein, so werden während der Extraktion Fehlermeldungen ausgegeben.
`-d`	Die Zeichenkette hinter dieser Option wird als File-Ende-Markierung anstelle von „SHAR_EOF“ gewählt.
`-p`	Das Zeichen hinter dieser Option ist das Präfix-Zeichen jeder zu extrahierenden Zeile eines Files. Anstelle des Kommandos **`cat`** wird der Stream-Editor **`sed`** zum Anlegen eines neuen Files verwendet.
`-s`	Die Extraktion erfolgt ohne Bildschirmausgabe. Es findet keine Überprüfung der zu extrahierenden Files statt.
`-v`	Während der Extraktion werden Meldungen ausgegeben.

In der Regel lassen sich die Optionen zu einer Zeichenkette `-`*`kette`* zusammenfassen, es sei denn, eine Option erfordert einen Parameter. In diesem Fall ist die Option einzeln und

durch ein Leerzeichen getrennt dahinter der Parameter anzugeben.

Als Beispiel sei die Umbenennung der File-Ende-Markierung gewählt. Soll die File-Ende-Markierung „Ende“ lauten, so ist die Utility **shar** folgendermaßen aufzurufen:

```
shar -d Ende filename > filename.shar
```

B.3 Man-Pages der Utility shar

Zusammen mit der Software für die Utility **shar** werden auch Man-Pages ausgeliefert.

Sie lassen sich mit Hilfe des Kommandos

```
man shar
```

am Bildschirm anzeigen. Mit den üblichen Umlenkmechanismen, zum Beispiel mit

```
man shar > shar.man
```

läßt sich die Ausgabe in ein File mit dem in diesem Beispiel gewählten Namen „shar.man“ umlenken. Danach kann der Inhalt des Files auch gedruckt werden.

B.4 Besonderheiten unter Unix-Derivaten

Besonderheiten unter den einzelnen Unix-Derivaten sind in der Regel nicht zu erwarten, da der Utility **shar** unter allen Derivaten dasselbe C-Programm zugrundeliegt.

Unter Umständen sind vor der Kompilation im C-Programm Modifikationen anzubringen, um Compiler-Besonderheiten zu berücksichtigen. In den untersuchten Unix-Derivaten sind jedoch nur teilweise geringfügige Änderungen erforderlich.

ANHANG C `compress` und `uuencode` zum Nachschlagen

Im folgenden soll zusammenfassend die allgemeine Syntax der Aufrufe der Kommandos **`compress`** und **`uuencode`** zum Komprimieren und Kodieren von Files dargestellt werden.

Für das Kommando **`compress`** sind noch andere Optionen als die im Kapitel „**`compress`** - zur Komprimierung von Files, **`uuencode`** - zur Kodierung von Files“ vorgestellte Option **`-d`** wählbar. Diese sind ebenfalls aufgeführt.

C.1 Allgemeine Syntax des Kommandos `compress`

Soll das Kommando **`compress`** zur Komprimierung eines Files verwendet werden, so darf die Option **`-d`** *nicht angegeben* werden.

Soll das Kommando **`compress`** zur Dekomprimierung eines komprimierten Files verwendet werden, so muß die Option **`-d`** *angegeben* werden.

C.1.1 Zur Komprimierung eines Files

Das File *`filename`* wird durch

`compress` `-`*`optionen filename`*

komprimiert. In der Zeichenkette *`optionen`* darf **`d`** nicht vorkommen. Andere Optionen `-`*`optionen`* sind wählbar.

Das Kommando erzeugt ein neues File *`filename`***`.Z`**, das alle Zugriffsrechte und das Veränderungsdatum des ursprünglichen Files *`filename`* trägt. Das File *`filename`* wird gelöscht

Anstelle von *filename* können auch mehrere Namen von Files *file1*, *file2*, ..., *filen* angegeben werden, die dann nacheinander komprimiert werden. Es ist nicht möglich, anstelle von *filename* den Namen einer Subdirectory einzusetzen.

C.1.2 Zur Dekomprimierung eines Files

Das komprimierte File *filename*.Z wird durch

compress -d*optionen filename*.Z

oder durch

uncompress -*optionen filename*.Z

wieder dekomprimiert. Weitere Optionen -*optionen* sind wählbar. Das Kommando erzeugt wieder das ursprüngliche File *filename*. Das File *filename*.Z wird gelöscht.

Anstelle von *filename*.Z können auch mehrere Namen von komprimierten Files *file1*.Z, *file2*.Z, ..., *filen*.Z angegeben werden, die dann nacheinander dekomprimiert werden.

C.2 Optionen des Kommandos compress

Das Kommando **compress** besitzt die in der folgenden Tabelle aufgeführten Optionen.

TABELLE 10 Optionen des Kommandos compress

Option	*Erläuterung*
-b	Die unmittelbar hinter dieser Option angegebene Zahl zwischen 9 und 16 ist die maximale Zahl der Bits, die zur Kodierung einer Zeichenkette verwendet werden sollen. Standardmäßig werden 16 Bits verwendet.

TABELLE 10 Optionen des Kommandos `compress`

Option	*Erläuterung*
`-c`	Die Ausgabe des Kommandos erfolgt auf die Standardausgabe. **`compress -cd`** entspricht dem Kommando **`zcat`**.
`-d`	Zur Dekomprimierung. Das Kommando **`compress -d`** entspricht dem Kommando **`uncompress`**.
`-f`	Komprimiert ein File auch dann, wenn das komprimierte File größer wird als das nicht komprimierte.
`-v`	Schreibt den Prozentsatz der Komprimierung auf die Standardausgabe.

Die Optionen **`-c`**, **`-d`**, **`-f`** oder **`-v`** können zu einer Zeichenkette der Form **`-`***`kette`* zusammengefaßt werden. Hinter der Option **`-b`** muß unmittelbar die Zahl der Bits folgen.

C.3 Man-Pages des Kommandos `compress`

Üblicherweise findet man unter den Unix-Derivaten die Erläuterungen zu einzelnen Kommandos in den Man-Pages. Um die genaue Beschreibung des Kommandos **`compress`** am Bildschirm lesen zu können, muß folgendes Kommando eingegeben werden:

```
man compress
```

Danach erscheint die ausführliche Beschreibung am Bildschirm. Mit den üblichen Umlenkmechanismen, zum Beispiel mit

```
man compress > compress.man
```

läßt sich die Ausgabe in ein File mit dem in diesem Beispiel gewählten Namen „compress.man“ umlenken. Danach kann der Inhalt des Files auch gedruckt werden.

C.4 Besonderheiten von `compress` unter einigen Unix-Derivaten

Besonderheiten des Kommandos **`compress`** unter einigen Unix-Derivaten sind nur insofern zu verzeichnen, als zusätzlich zu den in der Tabelle „Optionen des Kommandos **`compress`**“ aufgeführten Optionen noch weitere zur Verfügung stehen.

Die weiteren Optionen sind in den folgenden Tabellen erläutert.

C.4.1 IBM AIX

Unter dem Betriebssystem AIX stehen die folgenden weiteren Optionen des Kommandos **`compress`** zusätzlich zu den in der Tabelle „Optionen des Kommandos **`compress`**“ erwähnten zur Verfügung.

TABELLE 11 Weitere Optionen des Kommandos `compress` unter AIX

Option	*Erläuterung*
`-F`	Entspricht der Option **`-f`**.
`-n`	Läßt den komprimierten File-Header des komprimierten Files weg; zur Kompatibilität mit alten Versionen des Kommandos.
`-q`	Unterdrückt die durch die Option **`-v`** bedingte Ausgabe der Komprimierungsstatistik.

C.4.2 Linux

Unter dem Betriebssystem Linux stehen die folgenden weiteren Optionen des Kommandos **compress** zusätzlich zu den in der Tabelle „Optionen des Kommandos **compress**“ erwähnten zur Verfügung.

TABELLE 12 Weitere Optionen des Kommandos compress unter Linux

Option	*Erläuterung*
-C	Verhindert eine Aufteilung des Files in Blöcke; zur Kompatibilität mit alten Versionen des Kommandos.
-V	Die Versionsnummer des Kommandos wird ausgegeben.

C.4.3 SCO-Unix

Unter dem Betriebssystem SCO-Unix stehen die folgenden weiteren Optionen des Kommandos **compress** zusätzlich zu den in der Tabelle „Optionen des Kommandos **compress**“ erwähnten zur Verfügung.

TABELLE 13 Weitere Optionen des Kommandos compress unter SCO-Unix

Option	*Erläuterung*
-F	Entspricht der Option **-f**.
-q	Erzeugt kein komprimiertes File, sondern nur Fehlermeldungen, wenn Fehler vorkommen.

C.4.4 OSF/1

Unter dem Betriebssystem OSF/1 stehen die folgenden weiteren Optionen des Kommandos **compress** zusätzlich zu den in der Tabelle „Optionen des Kommandos **compress**“ erwähnten zur Verfügung.

TABELLE 14 Weitere Optionen des Kommandos compress unter OSF/1

Option	*Erläuterung*
-C	Verhindert eine Aufteilung des Files in Blöcke; zur Kompatibilität mit alten Versionen des Kommandos.
-F	Entspricht der Option **-f**.
-V	Die Versionsnummer des Kommandos wird ausgegeben.
-n	Läßt den komprimierten File-Header des komprimierten Files weg; zur Kompatibilität mit alten Versionen des Kommandos.
-q	Unterdrückt die durch die Option **-v** bedingte Ausgabe der Komprimierungsstatistik.

C.5 Allgemeine Syntax der Kommandos uuencode und uudecode

C.5.1 Zur Kodierung eines Files

Soll das File *filename* mit dem Unix-Kommando **uuencode** kodiert werden, so kann folgende allgemeine Aufrufsyntax verwendet werden:

```
uuencode filename dekod_file > filename.uu
```

Für *dekod_file* ist der Name des Files einzusetzen, das nach der Dekodierung den dekodierten Code enthalten soll. Wird *filename* in der Aufrufzeile weggelassen, so wird die Eingabe von der Standardeingabe erwartet. Erfolgt die Ausgabe nicht in ein File, zum Beispiel in *filename*`.uu`, so wird die Ausgabe des Kommandos auf die Standardausgabe geschrieben. Damit ist das Kommando **uuencode** auch für eine Pipe verwendbar. Für den Namen des kodierten Files wird auch manchmal die Endung **.uue** verwendet.

Das Kommando **uuencode** besitzt *keine Optionen*.

C.5.2 Zur Dekodierung eines Files

Ein mit **uuencode** kodiertes File *filename* wird mit Hilfe von **uudecode** folgendermaßen wieder dekodiert:

```
uudecode filename
```

Danach wird das dekodierte File erzeugt, dessen Name bei der Kodierung mit **uuencode** festgelegt wurde. Das File *filename* wird nicht verändert. Das Kommando **uudecode** ignoriert alle Einträge im zu dekodierenden File, die sich vor der Zeile, die mit „begin“ anfängt, und sich hinter der Zeile, die mit „end“ anfängt, befinden.

Das Kommando **uudecode** besitzt *keine Optionen*.

C.6 Man-Pages der Kommandos uuencode und uudecode

Üblicherweise findet man unter den Unix-Derivaten die Erläuterungen zu einzelnen Kommandos in den Man-Pages. Die Man-Pages der Kommandos **uuencode** und **uudecode** sind identisch, so daß es genügt,

```
man uuencode
```

einzugeben, um die genaue Beschreibung der Kommandos **uuencode** und **uudecode** am Bildschirm lesen zu können.

Mit den üblichen Umlenkmechanismen, zum Beispiel mit

```
man uuencode > uuencode.man
```

läßt sich die Ausgabe in ein File mit dem in diesem Beispiel gewählten Namen „uuencode.man“ umlenken. Danach kann der Inhalt des Files auch gedruckt werden.

C.7 Besonderheiten der Kommandos `uuencode` und `uudecode` unter Unix-Derivaten

Besonderheiten der Kommandos **`uuencode`** und **`uudecode`** unter den untersuchten Unix-Derivaten sind nicht vorhanden.

Index

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X

Z