Prof. Franz Schweiggert
Christian Ehrhardt
Abteilung Angewandte Informationsverarbeitung
2. Mai 2006
Blatt 2

Systemnahe Software (SS 2006)

Abgabetermin 09.Mai 2006

Eigene Archive

In diesem Blatt soll ein Programm geschrieben werden, mit dem sich viele verschiedene Dateien in einer einzigen Datei (einem sogenannten Archiv) zusammenfassen lassen. Natürlich soll es auch möglich sein, diese Dateien zu einem späteren Zeitpunkt wieder aus dem Archiv zu extrahieren. Dabei soll das Programm die folgenden Optionen unterstützen:

-f archivname Diese Option muß immer angegeben werden, sie gibt an, welche Datei als Archiv verwendet werden soll.

-c Ein neues Archive wird erstellt. Die in das Archiv zu verpackenden Dateien werden auf der Kommandozeile einzeln angegeben.

-l Mit dieser Option zeigt das Programm an, welche Dateien sich in dem Archiv befinden ohne es zu entpacken.

-e Diese Option entpackt das Archiv.

Der Einfachheit halber müssen nur zwei Dateitypen richtig verarbeitet werden können: Reguläre Dateien und Symbolische Links. Insbesondere müssen keine Verzeichnisse verarbeitet werden.
Ihr könnt davon ausgehen, daß sich die Dateien, die archiviert werden in dieser Zeit nicht ändern. Die einzige Ausnahme ist natürlich das Archiv selbst. Euer Programm sollte in der Lage sein festzustellen, ob das Archiv (ev. auch unter einem anderen Namen) selbst als zu archivierende Datei angegeben wurde und diese Datei ggf. mit einer Warnung überspringen.

Anleitung

Dateiformat

Bevor wir beginnen sollten wir uns darüber Gedanken machen, wie die notwendigen Informationen im Archiv abgelegt werden. Ich schlage folgendes vor:

- Die Dateien werden im wesentlichen einfach hintereinander in das Archiv geschrieben.
- Vor jeder Datei steht ein Header bestehend aus:
  - Ein Zeichen, das angibt, ob es sich bei der Datei um einen Symlink oder um eine reguläre Datei handelt.
  - Ein Integer, der die Länge des Datei**namens** angibt.
  - Ein Integer, der die eigentliche Länge der Datei bzw. des Symlinks angibt.
  - Anschließend folgt der Dateiname bestehen aus genau so vielen Zeichen, wie vorher angegeben.
  - Zum Schluß folgt der Inhalt der Datei. Im Falle eines symbolischen Links besteht dieser aus dem Ziel des Symlinks, also aus einem String, der angibt, wohin der Symlink verweist.

Darstellung von Zahlen

Aufgrund von immer wieder auftauchenden Fragen sei darauf hingewiesen, daß es zwei grundverschiedene Möglichkeiten gibt, um Zahlen darzustellen:

- Bei der normalen Darstellung einer Zahl im Rechner besteht ein `int` aus in der Regel 4 Bytes (genauer `sizeof(int)`). Mit dieser Darstellung kann der Rechner umgehen und arbeiten, sie ist aber für Menschen nur schwer lesbar.
- Für die Ausgabe hat man gerne eine Darstellung der Zahl als String bzw. als Folge von Ziffern. Dabei besteht die Zahl dann aus so vielen Ziffernzeichen, wie für Ihre Darstellung im Dezimalsystem benötigt werden. Daß es sich bei einem gegebenen String dann um die lesbare Darstellung einer Zahl handelt ist im wesentlichen eine Frage der (menschlichen) Interpretation. Insbesondere kann der Rechner mit Zahlen in dieser Darstellung nicht direkt Rechenoperationen ausführen.

Für den oben angesprochenen Header in unserem Archiv verwenden wir natürlich die erste der beiden Darstellungen, weil sie eine feste Größe hat. Eine Konvertierung (in diesem Blatt eigentlich nicht notwendig) erfolgt über die Funktionen der `scanf`-Familie in der einen und über die Funktionen der `printf`-Familie in der anderen Richtung.

Hinweis: Byte Ordnung

Beim Übertragen von solchen Archiven auf andere Rechner kann es zu Problemen kommen, weil sich die rechnerinterne Repräsentation von Zahlen zusätzlich unterscheidet (Stichwort Byte-Ordnung, mit diesem Problem werden wir uns erst später im Zusammenhang mit Netzwerken näher befassen).

Hauptprogramm

Wir beginnen dieses Mal mit dem Hauptprogramm und der Argumentverarbeitung. Falls noch nicht geschehen, schaut Euch dazu am besten die Manualseite von `getopt(3c)` an. Nachdem wir wissen, was wir tun wollen (neues Archiv erstellen oder ein bestehendes anzeigen/entpacken), wird die Archivdatei geöffnet und eine passende Prozedur aufgerufen. Beim Erzeugen eines neuen Archivs müssen natürlich die Kommandozeilenargumente mit übergeben werden, damit wir wissen welche Dateien eingepackt werden sollen.

Kopieren von Daten

Schreibt dann, damit wir das schon mal haben eine Prozedur `do_copy`, die eine genau vorgegebene Anzahl von Bytes aus einem Filedeskriptor liest und in einen anderen schreibt.

Archivieren einer Datei

Vor dem Archivieren muß natürlich festgestellt werden, ob es sich bei der Datei um einen Symlink oder eine reguläre Datei handelt. Hierbei hilft `lstat(2)`. Bei regulären Dateien bekommen wir so auch gleich die Länge der Datei und wir erfahren mit Hilfe der Inode-Nummer, ob es sich um das Archiv handelt oder nicht.
Anschließend wird zunächst der Header in die Datei geschrieben, gefolgt vom Namen der Datei. Dann folgt noch der Inhalt bzw. das Ziel des Symbolischen Links. Achtet hierbei strikt darauf, daß immer genau so viele Zeichen geschrieben werden, wie im Header für diese Datei angegeben. Im Falle einer regulären Datei kann die vorher geschriebene Prozedur `do_copy` zum Kopieren des Dateiinhalts verwendet werden. Bei einem Symlink bekommt man das Ziel des Symlinks mit Hilfe des Systemaufrufs `readlink(2)`.

Archivieren aller Dateien

Es empfiehlt sich, das Archivieren einer Datei so zu gestalten, daß der bereits geöffnete Filedeskriptor des Archivs übergeben wird. Die Datei wird dann samt Header genau beginnend an der aktuellen Position archiviert. In diesem Fall kann man einfach nacheinander alle Dateien archivieren ohne dazwischen die geöffnete Archivdatei manipulieren zu müssen.

Lesen eines Datei-Headers

Schreibt eine Prozedur `readheader`, die aus einem vorgegebenen Filedeskriptor an der aktuellen Position einen Header liest. Wenn er nicht vollständig gelesen werden kann oder wenn der Dateityp ungültig ist, sollte ein Fehler produziert werden. Wenn garnichts mehr gelesen werden kann, weil das Dateiende erreicht ist, wird ein NULL-Zeiger zurückgeliefert. Ansonsten wird eine Struktur zurückgeliefert, die den Dateityp, die Länge des Dateinamens, die Länge des Dateiinhalts und den Dateinamen enthält. Für die Struktur und den Dateinamen muß natürlich Speicher alloziert werden. Da der Dateiname im Archiv nicht durch ein Nullbyte terminiert wird, muß dies nach dem Lesen von Hand nachgeholt werden.

Der Inhalt eines Archivs

Es muß so lange ein Header gelesen werden, bis ein NULL-Zeiger zurückgeliefert wird. Für jeden Header wird dann ausgegeben, ob es sich um eine reguläre Datei oder einen Symlink handelt und wie die Datei heißt. Beide Informationen finden sich im gelesenen Header. Bei einer regulären Datei wird jetzt einfach mit `lseek` der Inhalt der Datei übersprungen, so daß der nächste Header gelesen werden kann. Bei einem symbolischen Link wollen wir natürlich sehen, wohin dieser zeigt. Dazu wird der Inhalt der Datei im Archiv, der ja bei symbolischen Links das Ziel des Links angibt gelesen und ebenfalls ausgegeben. In diesem Fall wird der Inhalt ja tatsächlich gelesen und deshalb natürlich nichts übersprungen.

Entpacken eines Archivs

Das entpacken funktioniert im Prinzip genau wie das Anzeigen des Inhalts. Es müssen nur die Dateien auch tatsächlich angelegt werden. Bei einem Symlink wird dazu die Funktion `symlink(2)` mit den gelesenen Daten als Parameter aufgerufen.
Bei regulären Dateien wird die neu anzulegende Datei geöffnet und statt dem Aufruf von `lseek` wird `do_copy` aufgerufen.

Fehlerbehandlung

Grundsätzlich ist auf eine sinnvolle Fehlerbehandlung zu achten.

Viel Erfolg