Lesen und Schreiben von ASCII-Files

ASCII- und Binärfiles

Unter *ASCII-*, *Text-* oder *formatierten Files* versteht man eine Form der Darstellung, bei der die gesamte in einem File enthaltene Information—egal, ob es sich um Text oder Zahlen handelt—unterschiedslos als Folge von "Zeichen" interpretiert wird, die in Form ihrer ASCII-Codes[1] gespeichert werden. Das hat den Vorteil, daß die Files direkt mit einem Editor oder Pager lesbar und relativ portabel sind. Andererseits müssen dabei auch Zahlen in Zeichenketten umgewandelt werden, was zu einem relativ hohen Platzbedarf führt. So benötigt man etwa zur Darstellung der Zahl `1.234567e+08` als Zeichenkette 12 Bytes, obwohl dieser Wert bequem in eine 32-Bit-Variable, also 4 Bytes, paßt. Dazu kommt, daß die Umwandlung von Floating Point-Zahlen in Zeichenketten (wegen des Übergangs zwischen binärer und dezimaler Darstellung) nicht verlustfrei ist: Schreibt man eine solche Variable auf ein File und liest sie wieder ein, so ist nicht garantiert, daß man Bit für Bit denselben Wert erhält.

Im Gegensatz dazu wird bei *unformatierten* oder *Binärfiles* ein direktes Abbild der Speicherinhalte auf das File geschrieben. Da dabei keine Konversion stattfindet, ist diese Form der Speicherung verlustfrei und wesentlich platzsparender. Allerdings sind Binärfiles nicht mehr mit "freiem Auge" (Editor) lesbar und wegen der Verwendung der maschinenspezifischen Zahlendarstellung auch wenig portabel. Ist insbesondere das Wissen um Typ und Reihenfolge der Daten einmal verlorengegangen, so kann es unmöglich sein, die im File enthaltene Information zu rekonstruieren.

Logische Einheiten und File-Pointer

Vor dem Lesen oder Schreiben eines Files muß vom Programm eine Zuordnung zwischen einem sprachspezifischen Identifikationsmechanismus und dem Filenamen im Betriebssystem hergestellt werden. Dieser Identifikationsmechanismus ist in F die *logische Einheit* (ein ganzzahliger Wert, mit dem ein "Kommunikationskanal" verknüpft wird), in C der *File-Pointer* (ein Pointer auf eine Struktur, in der Informationen über das File zusammengefaßt sind). Die Herstellung dieser Zuordnung nennt man das "Öffnen" eines Files. Danach können die eigentlichen Lese- oder Schreiboperationen auf dem File vorgenommen werden. Die Zuordnung zwischen logischer Einheit/File-Pointer und File hebt man auf, indem man das File "schließt". Die Reihenfolge ist also:

1. File öffnen
2. Lese- und/oder Schreiboperationen
3. File schließen.

[1] Nach dem American Standard Code for Information Interchange wird jedes Zeichen durch einen 1-Byte Code im Bereich 0–127 dargestellt. So sind z.B. die Codes der Kleinbuchstaben 'a', 'b', 'c' die Werte 97, 98, 99, die der Ziffern '0', '1', '2' die Werte 48, 59, 50. Neben ASCII gibt es auch noch eine Reihe anderer Codesysteme, vor allem für Texte und Zeichen in nicht-englischen Sprachen.

Files können zu beliebigen Zeitpunkten im Programm geöffnet und geschlossen werden. Es können mehrere Files gleichzeitig "offen" sein, zwischen denen dann durch die zugeordneten logischen Einheiten bzw. File-Pointer unterschieden wird.

Öffnen und Schließen von Files

In F öffnet man ein File durch den Befehl

`open(unit=`*unit*`,file=`*file*`,status=`*status*`,action=`*action*`[,...])`

Dabei sind die zwingend vorgeschriebenen Parameter

unit	logische Einheit (ganzzahliger Ausdruck)
file	Filename (Zeichenkettenausdruck)
status	`"old"`, `"new"` oder `"replace"`
action	`"read"`, `"write"` oder `"readwrite"`

Daneben gibt es noch eine Reihe optionaler Parameter wie `form="formatted"` (Standardannahme) oder `"unformatted"` und `position="rewind"` oder `"append"` (anfängliche Positionierung auf Beginn bzw. Ende des Files). Das File wird geschlossen mit

`close(unit=`*unit*`)`

Im Unterschied zu F, wo die logische Einheit beliebig wählbar ist (aber natürlich eindeutig sein muß), ist in C der File-Pointer durch den Rückgabewert der Funktion `fopen` gegeben und daher vom System bestimmt. Es muß also zunächst ein File-Pointer vom Typ `FILE` deklariert werden

`FILE *`*filepointer*`;`

Das File wird dann geöffnet mit

filepointer`=fopen(`*file*`,`*mode*`);`

Hier ist der Zeichenkettenausdruck *file* wieder der Filename, und *mode* bezeichnet den Zugriffsmodus mit den möglichen Werten

`"r"`	Lesen
`"r+"`	Lesen und Schreiben
`"w"`	Schreiben (existierendes File wird gelöscht)
`"w+"`	Lesen und Schreiben (existierendes File wird gelöscht)
`"a"`	Schreiben (Positionierung auf File-Ende)
`"a+"`	Lesen und Schreiben (Positionierung auf File-Ende)

Das File wird geschlossen mit

`fclose(`*filepointer*`);`

Lese- und Schreibbefehle

Die eigentlichen Lese- und Schreibbefehle für formatierte Files sind in F genau dieselben wie für die Ein/Ausgabe auf Tastatur und Bildschirm, nur wird statt `"unit=*"` jetzt die logische Einheit angegeben, mit der das File geöffnet wurde. Das heißt also für Lesen von einem File mit der zugeordneten logischen Einheit *unit*

`read(unit=`*unit*`,fmt="(`*format*`)")` *variable_list*

und für Schreiben auf ein File

`write(unit=`*unit*`,fmt="(`*format*`)")` *expression_list*

In C treten an die Stelle von `scanf` und `printf` die Funktionen `fscanf` und `fprintf`, die als zusätzliches erstes Argument den dem File zugeordneten File-Pointer enthalten, sonst aber vollkommen analog zu `scanf` und `printf` sind. Man liest also in C von einem formatierten File mit

`fscanf(`*filepointer*`,"`*format*`",`*variable_list*`);`

und schreibt mit

`fprintf(`*filepointer*`,"`*format*`",`*expression_list*`);`

Das Lesen und Schreiben von Files erfolgt grundsätzlich sequentiell. In beiden Sprachen gibt es aber noch zusätzliche Befehle zur genaueren Positionierung auf Files. F kennt außer sequentiellen Files auch noch (bei Binärfiles) den Typ des Direktzugriffsfiles.