Grundlagen der Dateibehandlung

Dieses Hilfsblatt beschreibt einige wichtige POSIX Routinen zur Dateibehandlung, etwa in dem Umfang, wie sie zur Bearbeitung der anstehenden Aufgabe benötigt werden.

Grundsätzlich bieten Unix-basierte Systeme die Möglichkeit, entweder über ein *high-level* oder *low-level* Interface Dateizugriffe durchzuführen. Das high-level Interface arbeitet mit *Datenströmen* vom Typ **`FILE`**. Z. B. arbeiten die Funktionen zur formatierten Ausgabe (z. B. **`printf`**) mit Zeigern auf **`FILE`**-Objekte.

Innerhalb der Prozesse werden die jeweils Verweise auf die verwendeten Dateien in einer Tabelle verwaltet, wobei die Indizes als sog. *Dateideskriptoren* bezeichnet werden. Da die Deskriptoren Integer-Werte ohne zusätzliche Information darstellen, entspricht dieses Konzept also einer low-level Schnittstelle.

Einträge in der Deskriptor-Tabelle verweisen auf Meta-Daten einer vom Kernel verwalteten Dateitabelle. Durch dieses Konzept können auf recht einfache Weise von unterschiedlichen Prozessen auf gleiche Dateien zugegriffen werden, ohne dass die Meta-Daten kopiert werden müssen. So können z. B. Pipes einfach erstellt und Verweise auf Dateien kopiert werden.

Die beiden Funktionen

- **`int fileno (FILE* stream)`** und
- **`FILE* fdopen (int fd, const char* mode)`**

ermöglichen die Ermittlung eines Deskriptors zu einem Zeiger auf ein **`FILE`**-Objekt und umgekehrt.

Öffnen oder Erzeugen einer Datei

Syntax

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>  /* fuer Kontrollfluss */

int open(const char *path, int oflag,… /* mode_t mode */);
```

Parameter

Die wichtigsten Parameter sind:

- **`path:`** Name der Datei, entweder voll qualifiziert oder mit relativem Pfad
- **`oflag`**: Bitverknüpfung von Flags (insgesamt 12 definiert). z. B.:
  - **`O_RDONLY`**: öffnet die Datei nur zum Lesen.
  - **`O_WRONLY`**: öffnet die Datei nur zum Schreiben.
  - **`O_RDWR`**: Schreiben und Lesen erlaubt.
    (nur eines dieser 3 Flags ist erlaubt!)
  - **`O_APPEND`**: geschriebene Daten werden ans Ende einer Datei angehängt.

- **O_CREAT**: Falls die Datei nicht existiert, wird sie erzeugt (**mode** Parameter muss übergeben werden.
  ….
- **mode** (optional): Legt die Zugriffsrechte beim Erzeugen einer Datei fest.
  Zur Definition der **mode_t** Flags siehe z. B. **man -s 2 mknod** .

Rückgabewert

- Wenn erfolgreich: positiver Wert, der den **Dateideskriptor** repräsentiert.
  Reservierte Werte:
  - **STDIN_FILENO** (0): **stdin**, Standardeingabe
  - **STDOUT_FILENO** (1): **stdout**, Standardausgabe
  - **STDERR_FILENO** (2): **stderr**, Standardfehlerausgabe

  open liefert jeweils den kleinsten freien Dateideskriptor zurück, also Werte ab Deskriptor 3.
- Im Fehlerfall: -1. **errno** kann zur Fehlerdiagnose benutzt werden.

Schließen einer Datei

Syntax

```
#include <unistd.h>
int close(int fd);
```

Schließt eine geöffnete Datei nach deren Benutzung.

Parameter

- **fd:** Deskriptor der geöffneten Datei, die geschlossen werden soll.

Rückgabewert

- 0 wenn erfolgreich, sonst -1. **errno** kann zur Fehlerdiagnose genutzt werden.

Lesen aus seiner Datei

Syntax

```
#include <unistd.h>
ssize_t read(int fd, void *buf, size_t nbytes);
```

Liest maximal **nbytes** Bytes aus **fd** in den Zeichenpuffer **buf** ein. Der Aufrufer ist dafür verantwortlich, dass **buf** auf einen genügend großen freien Speicherbereich zeigt.

Rückgabewert

- Die Zahl der gelesenen Bytes oder
- -1 im Fehlerfall.

Schreiben in eine Datei

Syntax

```
#include <unistd.h>
ssize_t write(int fd, const void *buf, size_t nbytes);
```

Schreibt maximal **nbytes** Bytes aus **buf** in **fd** ein.

Rückgabewert

- Die Anzahl der geschriebenen Bytes.
- Wird **write** unterbrochen, kann die Anzahl ungleich **nbytes** sein! **errno** gibt Auskunft darüber (**EINTR**)!

Positionieren des Dateizeigers

Syntax

```
#include <unistd.h>
off_t lseek(int fd, off_t *offset, int whence);
```

Positioniert den Dateizeiger auf die Stelle, ab der aus der Datei gelesen oder in die Datei geschrieben werden soll.
Nicht alle Dateideskriptoren erlauben das freie Positionieren des Dateizeigers. Aus- bzw. Eingabeströme wie z. B. **STDOUT_FILENO** und **STDIN_FILENO** oder Ströme, die zu zeichenorientierten Geräten gehören, erlauben kein Umsetzen.

Parameter

- **fd**: Deskriptor einer geöffneten Datei.
- **offset**: Der Dateizeiger wird um **offset** Bytes ab **whence** verschoben.
- **whence**:
  - **SEEK_SET** : **offset** gibt die absolute Position in der Datei an (ab Dateianfang).
  - **SEEK_CUR**: **offset** wird ab der aktuellen Position gezählt.
  - **SEEK_END**: **offset** zählt ab dem Dateiende.

Rückgabewert

- Die absolute Position (ab Dateianfang) in der Datei.
- Im Fehlerfall wird (**off_t**) -1 zurückgegeben.

Abfrage des Dateizeigers

Syntax

```
#include <stdio.h>
long ftell (FILE* stream);
```

Gibt die aktuelle Position des Dateizeigers zurück.

Parameter

- **stream**: Verweis auf Datei.

Rückgabewert

- Die absolute Position (ab Dateianfang) in der Datei.

Referenzen

- Steve Grägert: POSIX-Programmierung mit UNIX, siehe Stud.IP
- W. Richard Stevens, Stephen A. Rago: Advanced Programming in the UNIX Environment. Second Edition, Addison-Wesley Professional, 2008. ISBN 0321525949
- Helmut Weber: Praktische Systemprogrammierung. Vieweg 1998. ISBN 3528056584.
- B. O. Gallmeister: POSIX.4 Programming for the Real World. O'Reilly Sebastopol, 1995. ISBN 1565920740.
- Horn, Thomas: Systemprogrammierung. Verlag Technik Berlin, 1994. ISBN 3341010904.
- M. Mitchell, J. Oldham, A. Samuel: Advanced Linux Programming. New Riders Publishing; 2001 (Online verfügbar: http://www.advancedlinuxprogramming.com/alp-folder)