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. Diese werden vom Betriebssystem genutzt, um höherwertige Funktionen und Kommandos auf Shell-Ebene (z.B. das Kopieren von Dateien auf einem Dateisystem) zu realisieren. Die Basisroutinen laufen dabei im Kernel-Mode und haben deshalb vollen Zugriff auf Betriebsmittel und die damit verbundenen Daten. Im POSIX-Standard sind diese Routinen (und nicht die Systemaufrufe) spezifiziert, die einen Wrapper für die Systemaufrufe darstellen. Die Abarbeitung des Aufrufs einer Routine wird über Interrupts initiiert:

1. die Register des Kernels wird im Stack gesichert,
2. anschließend wird die Wrapper Funktion (auch Service-Routine genannt) ausgeführt und
3. schließlich wird einer Rückgabewert (ggf. Fehlercode) erzeugt. Die errno-Variable wird dabei durch die Wrapperfunktion beschrieben.

Im Folgenden wird eine Auswahl der in Frage kommenden POSIX Routinen gegeben.

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

Einige der 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 nur zum Lesen.
  - `O_WRONLY`: öffnet 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 Zugriffsrechte beim Erzeugen einer Datei fest. Zur Definition der `mode_t Flags siehe z.B. \verb`man –s 2 mknod!.

Prof. Dr.-Ing. Martin Hoffmann
martin.hoffmann@fh-bielefeld.de

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 den kleinsten freien Dateideskriptor zurück, damit Werte ab Deskriptor 3.
- Im Fehlerfall: -1. errno kann zur Fehlerauswertung benutzt werden.

Schließen einer Datei `Syntax`

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

Schließt eine geöffnete Datei nach deren Benutzung.

Parameter

- `fildes`: Deskriptor einer geöffneten Datei, die geschlossen werden soll.

Rückgabewert

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

Lesen

Syntax

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

Liest maximal `nbytes` Bytes aus `filedes` in `buf` ein. Der Aufrufer ist dafür verantwortlich, dass buf auf einen genügend großen freien Speicher zeigt.

Rückgabewert

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

Schreiben

Syntax

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

Schreibt maximal `nbytes` Bytes aus `buf` in `filedes` ein.

Rückgabewert

- Die Zahl der geschriebenen Bytes.
- Wird write unterbrochen, kann das ungleich nbytes sein! errno gibt Auskunft (EINTR)!

Positionieren des Dateizeigers

Syntax

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

Positioniert den Dateizeiger auf die Stelle, ab der aus einer Datei gelesen oder in eine Datei geschrieben wird. Nicht alle Dateideskriptoren erlauben ein positionieren des Dateizeigers. Zeichenorientierte Ausbzw. Eingabeströme wie z.B. `STDOUT_FILENO` und `STDIN_FILENO` erlauben kein Umsetzen.

Parameter

- `fildes`: 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.

Referenzen

- Steve Graegert: POSIX-Programmierung mit UNIX, siehe Ilias
- W. Richard Stevens, Stephen A. Rago: Advanced Programming in the UNIX Environment. Second Edition, Addison-Wesley Professional, 2008.
- Helmut Weber: Praktische Systemprogrammierung. Vieweg 1998.
- M. Mitchell, J. Oldham, A. Samuel: Advanced Linux Programming. New Riders Publishing; 2001