Allgemeine Hinweise zur Systemprogrammierung

Hier wird auf einige Dinge eingegangen, auf die im Rahmen des Praktikums besonders geachtet wird:

Posix Befehle:

siehe z.B.:

- Steve Grägert: Posix-Programmierung mit UNIX
  (in Stud.IP oder direkt: http://www.graegert.com/
  http://digitalether.de/index.php?option=com_content&task=view&id=52&Itemid=53)
- Beschreibung aller Funktionen:
  http://www.opengroup.org/onlinepubs/009695399/functions/contents.html
- man-Pages des Rechners.

Fehlerbehandlung:

Jede Posix-Funktion liefert einen Fehlercode zurück! Der Fehlergrund wird in der globalen Variablen `errno` gespeichert.

Im Rahmen des Praktikums muss jeder Systemaufruf auf Fehler überprüft werden!

Beispiel: `open` – Systemaufruf zum Öffnen einer Datei.

Syntax:

```
int fd = open ("Not Existing File", O_RDONLY);
```

Die Datei “Not Existing File” existiert nicht und kann somit auch nicht zum Lesen geöffnet werden.

Beim Scheitern des Befehls wird der Rückgabewert (fd), der ansonsten einen gültigen Dateideskriptor enthalten würde, auf -1 gesetzt.

Die Variable `errno` wird im Fall der nicht existierenden Datei auf den Wert `ENOENT` gesetzt.

Weitere Gründe für einen Fehler könnten sein (fd = -1):

- EACCES: keine Berechtigung, den Pfad, in dem die Datei geöffnet werden soll, zu durchsuchen.
- EMFILES: Die maximale erlaubte Anzahl an offenen Dateien ist erreicht.
- EINTR: Ein Signal wurde empfangen.
- ……

Insgesamt existieren 24 mögliche Gründe für das Scheitern des Befehls, die in `errno` gespeichert sein können.

Der Fehlergrund kann mittels der Funktion perror als Text auf `stderr` ausgegeben werden.

Syntax:

```
#include <stdio.h>

void perror(const char *s);
```

Für das Beispiel könnte die Behandlung wie folgt aussehen:

```
…
fd = open ("Not Existing File", O_RDONLY);
if (-1 == fd)
{
  perror("open");
  exit(-1);
}
…
```

Die Ausgabe auf `stderr` wäre:

```
open: No such file or directory
```

Die Fehlermeldung (hier „: No such file or directory“) wird von der `perror()` Funktion anhand des Wertes von `errno` bestimmt. Der Präfix (hier: „open“) ist der Übergabeparameter der `perror()` Funktion.

Bei einigen Systemaufrufen kann der Rückgabewert nicht zur Fehlerdetektion genutzt werden, da der Wertebereich vollständig für gültige Werte benutzt werden kann.

In diesem Fall muss `errno` vor dem Systemaufruf auf 0 gesetzt werden, da die Systemaufrufe den Wert von `errno` im Erfolgsfall nicht verändern. Nach dem Systemaufruf muss `errno` auf einen Wert ungleich 0 geprüft werden.

Beispiel: (hier wird `die readdir()` Funktion benutzt, deren Bedeutung ist erst einmal nicht wichtig)

```
#include <errno.h> /* Damit errno bekannt ist */
…
errno = 0;
sdir=readdir(dp);
if (0 != errno)
{
  perror("readdir");
  return errno;
}
…
```

Wie genau die Fehler erkannt werden können, ist der entsprechenden Beschreibung der Systemfunktionen zu entnehmen.

Ihre eigenen Fehlermeldungen sollten grundsätzlich auf `stderr` erfolgen!

```
fprintf(stderr, "My Error Message");
```