Übungsblatt 2 – Dateizugriffe

Aufgabe 1: Schauen Sie sich noch einmal alle Folien aus der Vorlesung an und stellen Sie sicher, dass Sie sie verstanden haben. Falls in der Vorlesung nicht alle Folien besprochen wurden, dann arbeiten Sie die nicht besprochenen Folien selbstständig durch. Benutzen Sie dabei auch die Literatur. Falls Fragen offen bleiben sollten, besprechen Sie diese während der Übungsstunde mit dem Betreuer.

Aufgabe 2:

a) Lesen Sie das Kapitel über (low-level) Dateizugriffe in dem Buch, das Sie sich besorgt haben. Die Abschnitte, in denen es in deutlich tiefere Details geht als in der Vorlesung angesprochen, können Sie überspringen.

b) Erstellen Sie Ihre persönliche handschriftliche Zusammenfassung der wichtigsten Inhalte der Lehrveranstaltung.

c) Überlegen Sie sich drei Verständnisfragen zur Lehrveranstaltung und halten diese schriftlich fest.

Wichtige Hinweise:

- Schreiben Sie alle Übungsprogramme vernünftig formatiert und sinnvoll mit Kommentaren dokumentiert.
- Bevor Sie einen Systemaufruf programmieren, lesen Sie in der Dokumentation der C-Library die Beschreibung dieses Systemaufrufs nach.
- Schreiben Sie alle folgenden C-Programme so, dass bei jedem Systemaufruf überprüft wird, ob ein Fehler aufgetreten ist. Falls Fehler auftreten, dann geben Sie eine aussagekräftige Fehlermeldung aus.
- Schalten Sie beim übersetzen immer alle Compilerwarnungen an! Benutzen Sie außerdem den Debug-Modus, so dass Sie ggf. mit einem Debugger nachvollziehen können, was ihr Programm macht.
- Automatisieren Sie alle Übersetzungsvorgänge mit Hilfe eines `Makefile` Fügen Sie ein target `clean` ein, das beim Aufruf alle Zwischen-, Hilfs- und erzeugte Programmdateien wieder löscht.

Aufgabe 3: Programmzugriff auf den file descriptor (FD). Lesen Sie in der Dokumentation der C-Library den Abschnitt 13.1 „Opening and Closing Files“.

a) Schreiben Sie ein C-Programm. das
   - Eine Datei im Read-Only-Modus öffnet.
   - Den Dateideskriptor als Dezimalzahl ausgibt
   - Dann eine Minute wartet und nichts macht. Benutzen Sie hierfür die Funktion `sleep`

b) Compilieren und starten Sie das Programm und ermitteln Sie die PID dieses Prozesses. Benutzen Sie hierfür das Kommando `ps`.

c) Schauen Sie sich nun den Verzeichnisinhalt von `/proc/<PID>/fd` an und vergleichen es mit der Ausgabe Ihres Programmes. Ersetzen Sie `<PID>` im Verzeichnisnamen durch die in Teilaufgabe b) ermittelte Prozessnummer.

Aufgabe 4: Erstellen Sie mit einem Editor eine Datei, in die Sie die Wörter „eins zwei drei vier“ schreiben. Lesen Sie in der Dokumentation der C-Library den Abschnitt 13.2 „Input and Output Primitives“.

a) Schreiben Sie ein C-Programm. das
   - Die gerade erstellte Datei im Read-Write-Modus öffnet.
   - Schreiben Sie mit dem `write`-Kommando das Wort „Hallo“ in die Datei.

b) Compilieren und starten Sie das Programm.

c) Schauen Sie sich nun den Inhalt der Datei an.

d) Kopieren Sie das C-Programm und ändern es wie folgt:
   - Die gerade erstellte Datei wird Read-Only-Modus öffnet.
   - Schreiben Sie mit dem `write`-Kommando das Wort „Hallo“ in die Datei.

e) Compilieren und starten Sie das Programm.

Aufgabe 5:

a) Schreiben Sie ein C-Programm, das
   - Eine Datei im Read-Write Modus öffnet
   - 30 000 mal „Hallo“ in die Datei schreibt. Hierfür können Sie das Kommando `write` verwenden
   - Die Datei wieder schließt

b) Compilieren und starten Sie das Programm. Messen Sie die Laufzeit mithilfe des Tools `time`.

b) Ändern Sie das Programm und geben den Parameter `O_SYNC` beim Öffnen mit an. Compilieren und starten Sie das Programm erneut. Messen Sie die Laufzeit mithilfe des Tools `time`. Erklären Sie sich die Unterschiede.

Aufgabe 6:

a) Schreiben Sie ein C-Programm, das
   - Eine Datei mit O_RDWR, O_CREAT und O_EXCL Modus öffnet
   - „Hallo“ in die Datei schreibt. Hierfür können Sie das Kommando `write` verwenden
   - Die Datei wieder schließt

b) Compilieren und starten Sie das Programm
   Erster Start: Zu schreibende Datei existiert noch nicht
   Zweiter Start: Zu schreibende Datei existiert bereits

c) Können Sie die erzeugte Datei lesen? Wenn nein, warum nicht?

Aufgabe 7:

a) Schreiben Sie ein C-Programm, das
   - Eine Datei im Read-Only-Modus öffnet
   - Schrittweise den Inhalt der Datei in einen Buffer liest (benutzen Sie hierfür `read`) und den Buffer dann auf die Standard-Ausgabe ausgibt. (hierfür können Sie `printf` benutzen).
   - Die Datei wieder schließt

b) Compilieren und starten Sie das Programm.

Aufgabe 8: Lesen Sie in der Dokumentation der C-Library den Abschnitt 13.3 „Setting the File Position of a Descriptor“.

a) Schreiben Sie ein C-Programm, das
   - Eine Datei im Read-Only-Modus öffnet
   - An das Ende der Datei springt und dann die Position des Zugriffszeigers ausgibt (benutzen Sie hierfür `lseek`)
   - Die Datei wieder schließt

b) Compilieren und starten Sie das Programm.

Aufgabe 9:

a) Schreiben Sie ein C-Programm, das
   - Eine Datei im Read-Write- und Append-Modus öffnet
   - An den Anfang der Datei springt und dort das Wort „Hallo“ schreibt. Ermitteln Sie die Position des Zugriffszeigers und geben Sie aus.
   - Die Datei wieder schließt

b) Compilieren und starten Sie das Programm.

c) Macht es einen Unterschied, ob die Zieldatei bereits existiert oder nicht?