Übungsblatt 3 – Buffered I/O und Directory-Zugriffe

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 Buffered I/O, Zugriff auf Statusinformationen von Dateien und Zugriffe auf Directories 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: Fehlercodes und Erklärungstext ausgeben.

a) Schreiben Sie ein C-Programm, das
   - Für jeden der Fehlercodes von 1 bis 99 den Code und den zugehörigen Erklärungstext ausgibt.

b) Compilieren und starten Sie das Programm.

Aufgabe 4: Buffered I/O und streams. Lesen Sie in der Dokumentation der C-Library die Abschnitte 12.7 bis 12.9 „Simple Output by Characters or Lines“ bis „Line-Oriented Input“.

a) Schreiben Sie ein C-Programm, das
   - Jedes Zeichen aus dem Standard-Input-Stream einzeln liest (fgetc) und dann einzeln in den Standard-Output-Stream schreibt (fputc).

b) Compilieren Sie ihr Programm und testen es. Messen Sie mit dem `time` Kommando die Laufzeit ihres Programmes, wenn sie eine ca. 20 MB große Textdatei (keine Binärdatei!) als Eingabe nehmen und die Standard-Ausgabe in eine Datei umlenken.

c) Schreiben Sie ein weiteres C-Programm, das
   - Jedes Zeile aus dem Standard-Input-Stream einzeln liest (fgets) und dann einzeln in den Standard-Output-Stream schreibt (fputs).

d) Compilieren Sie ihr Programm und testen es. Messen Sie mit dem `time` Kommando die Laufzeit ihres Programmes, wenn sie eine ca. 20 MB große Testdatei als Eingabe nehmen und die Standard-Ausgabe in eine Datei umlenken.

e) Welche Laufzeitunterschiede stellen Sie fest?

Aufgabe 5: Zugriff auf Dateiattribute. Lesen Sie in der Dokumentation der C-Library den Abschnitt 14.9 „File Attributes“.

a) Schreiben Sie ein C-Programm, das
   - den Namen einer Datei oder eines Verzeichnisses als Parameter erwartet.
   - mit einer Fehlermeldung abbricht, falls kein gültiger Name übergeben wurde.
   - Bei einem gültigen Namen ausgibt, um welche Art von Datei es sich handelt.

b) Compilieren und starten Sie das Programm.

c) Testen Sie Ihr Programm, indem Sie sich mindestens vier Testfälle überlegen und diese dann ausführen.

Aufgabe 6: Verzeichnis durchlaufen und Inhalt ausgeben. Lesen Sie in der Dokumentation der C-Library den Abschnitt 14.2 „Accessing Directories“.

a) Schreiben Sie ein C-Programm, das
   - den Namen eines Verzeichnisses als Parameter erwartet.
   - mit einer Fehlermeldung abbricht, falls kein gültiger Name übergeben wurde.
   - den Inhalt der Verzeichnisses darstellt und zu jedem Eintrag angibt, um welche Art von Datei es sich handelt.

b) Compilieren und starten Sie das Programm.

c) Erstellen Sie zwei Testverzeichnisse, eines leer und eines mit mindestens drei Einträgen unterschiedlicher Art. Testen Sie Ihr Programm mit diesen beiden Verzeichnissen.