

Programmieren 1 Übung: Strukturen

Klaus Kusche

1.) Textfile sortieren mit Zeilennummern

Wir erweitern die Aufgabe aus der **qsort**-Übung (du kannst deine Lösung oder meine *Musterlösung* für die **qsort**-Übung als Ausgangsbasis für diese Übung verwenden): In der Ausgabe soll vor jeder Zeile ihre ursprüngliche Zeilennummer im Eingabefile angezeigt werden.

Lösungsidee:

- Wir ersetzen unser ursprüngliches, dynamisch wachsendes Array von **char**-Pointern auf die Zeilentexte durch ein Array von Strukturen (eine Struktur pro Zeile).
- Diese Struktur soll zwei Member enthalten:
 - Wie bisher: Den **char-Pointer auf den Text der Zeile**.
 - Die Zeilennummer der Zeile im Eingabefile.
- Die Zeilennummer wird beim Einlesen der Zeile mitgespeichert und beim Ausgeben des sortierten Ergebnisses ausgegeben.

Hinweise:

- Verwende für den Strukturtyp wenn möglich ein **typedef**.
- Weiters musst du an folgenden Stellen ändern (und wenn möglich an keinen anderen Stellen!):
 - In der Vergleichsfunktion für **qsort**.
 - Im **qsort**-Aufruf (Größenangabe eines Elementes).
 - In der Deklaration des Pointers (oder der Pointer) auf das Array.
 - Im **malloc** und **realloc** (Größenberechnung und Typumwandlung des Ergebnisses).
 - Beim Einlesen und beim Ausgeben einer Zeile.

2.) Directory-Listing

Gesucht ist ein Programm, das ähnlich wie "**dir**" im DOS-Fenster oder "**ls**" in Linux Informationen über die Files in einem Verzeichnis ausgibt.

Realisiere dein Programm in drei Schritten:

1. Ausgabe nur der Filenamen im aktuellen Verzeichnis (Aufruf des Programms ohne Argumente auf der Befehlszeile).
2. Ausgabe der Filenamen im aktuellen Verzeichnis mit Zusatzinformation: File-Länge in Bytes, File-Änderungsdatum und File-Typ, dieser kann "f" (File), "d" (Verzeichnis) oder "?" (alles andere) sein.

3. Ausgabe wie in 2. für ein auf der Befehlszeile angegebenes Verzeichnis.
(es können nur Verzeichnisse angegeben werden, keine Files oder Wildcards)

Hinweise:

- Wir verwenden für dieses Programm die Posix- (Linux-) Funktionen für Directories. Diese stehen für Windows rudimentär auch unter DevCpp zur Verfügung (nicht aber unter Visual Studio!).

Du wirst dir die Details zu diesen Funktionen von den entsprechenden Man-Pages am Internet holen müssen.

Mach dir keine Sorgen, wenn dein Programm unter Windows nicht mit allen Kombinationen von Laufwerks- und Verzeichnis-Angaben auf der Befehlszeile zurechtkommt. Eine korrekte Behandlung aller Fälle wäre viel zu aufwändig...

- Unter Unix / Linux ist die Directory-Information in zwei getrennten Bereichen auf der Platte gespeichert. Dementsprechend gibt es auch zwei getrennte Funktionen, um diese Information abzurufen:
 - Im Verzeichnis selbst steht nur der Filename und die dazugehörige interne Filenummer. Um ein Verzeichnis Eintrag für Eintrag sequentiell zu lesen, gibt es die Funktionen **opendir** / **readdir** / **closedir** aus **dirent.h**. (Achtung: Nicht mit dem Linux System Call **readdir** verwechseln, die Hilfe zur Funktion ist **man 3 readdir**, die zum System Call **man 2 readdir**!).
 - In einem Systembereich der Platte steht zu jeder Filenummer die restliche Information. Sie wird u.a. mit der Funktion **stat** aus **sys/stat.h** abgerufen.
sys/stat.h definiert auch zahlreiche Makros, um das Ergebnis von **stat** auszuwerten, z.B. **S_ISREG** und **S_ISDIR**, um den Typ des Files zu prüfen.

Beide liefern verschiedene Strukturen als Ergebnis, die **struct**-Deklarationen stehen ebenfalls im jeweiligen Headerfile.

Achtung:

- In einem Fall ist der Returnwert ein Zeiger auf eine interne statische Struktur der Funktion.
- Im anderen Fall musst du eine leere Struktur für das Ergebnis als Argument übergeben!
- Die Umwandlung des internen File-Datums in Textform erfolgt am einfachsten mit der Funktion **ctime** (aus **time.h**). Diese Funktion liefert einen Datums- und Zeit-String mit einem **\n** hinten dran, das **\n** schneidest du vor der Ausgabe am besten weg.
- Die maximale Länge von Filenamen und Verzeichnispfaden (incl. **\0**) ist durch **PATH_MAX** aus **limits.h** definiert.
- In Version 1 und 2 machst du das **opendir** mit **"/"**, und das **stat** kannst du direkt mit dem Filenamen aufrufen, den **readdir** geliefert hat.

In Version 3 machst du das **opendir** mit dem angegebenen Pfad, und den Namen für das **stat** musst du aus dem angegebenen Pfad, einem '/' und dem Ergebnis von **readdir** händisch zusammenbasteln (auf die maximale Länge achten!).