

AIK Programmieren 1 Übung: Strukturen

Klaus Kusche

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 (siehe unten):

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 Verzeichnis-Namen angegeben werden, keine Filenamen und keine Wildcards mit *)

Hinweise:

- Wir verwenden für dieses Programm die Linux-Funktionen für Verzeichnisse. Diese stehen für Windows rudimentär auch unter DevCpp, CodeLite, Geany usw. zur Verfügung (aber nicht unter Visual Studio!).
- Unter Unix / Linux ist die Verzeichnis-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** und **readdir** aus dem Header **dirent.h**.
 - In einem Systembereich der Platte steht zu jeder Filenummer die **restliche Information** (Filetyp, Länge, Datum, Rechte & Besitzer, ...). Sie wird mit der Funktion **stat** aus dem Header **sys/stat.h** abgerufen. **sys/stat.h** definiert auch einige Funktionen, um das Ergebnis von **stat** auszuwerten, z.B. **S_ISREG** und **S_ISDIR**, um den Typ des Files zu prüfen.

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

Schritt 1:

- Bevor man Verzeichnis-Einträge lesen kann, muss man für das Verzeichnis **opendir** aufrufen: Hinein geht der Verzeichnis-Name (ein **char ***, in unserem Fall “.”, weil wir ja den Inhalt des aktuellen Verzeichnisses ausgeben wollen), und zurück kommt ein Pointer auf einen DIR-Wert. **DIR** ist ein mittels **typedef** in **dirent.h** definierter **struct**-Typ, was sich dahinter wirklich verbirgt, kann uns egal sein.

Wenn das **opendir** *schiefgeht*, kommt ein **NULL**-Pointer zurück.
In diesem Fall solltest du eine *Fehlermeldung* ausgeben und das Programm *beenden*.

- Mit dem **DIR**-Pointer aus **opendir** als Argument ruft man in einer Schleife immer wieder **readdir** auf.

readdir liefert bei jedem Aufruf den *nächsten* Verzeichnis-Eintrag zurück, und zwar in einer *Struktur* vom Typ **struct dirent** :

readdir enthält intern eine solche *statische* Struktur, trägt das Ergebnis dort ein, und liefert einen Pointer auf diese Struktur als Returnwert.

Geht das **readdir** schief (weil schon alle Einträge gelesen sind oder weil irgendein Fehler aufgetreten ist), kommt ein **NULL**-Pointer zurück; in diesem Fall beenden wir das Programm ganz normal.

Doku zu **readdir**:

<http://www.kernel.org/doc/man-pages/online/pages/man3/readdir.3.html>

- Uns interessiert in dieser Struktur **struct dirent** nur ein Member: **d_name** (ein **char**-Array), der *Filename*. Diesen String geben wir jedesmal aus.

Schritt 2:

- Für die File-Information rufen wir zusätzlich die Funktion **stat** auf.
Erstes Argument ist der Filename (so, wie ihn **readdir** in **d_name** geliefert hat), für den wir die Information haben wollen.

Heraus kommt eine **struct stat**, allerdings diesmal *nicht als Returnwert*:

Wir müssen in unserem Programm selbst eine Struktur-Variable mit dem vordefinierten Struktur-Typ **struct stat** für das Ergebnis *anlegen* und dem **stat** einen *Pointer auf unsere Struktur als zweites Argument* übergeben, **stat** trägt dann sein Ergebnis in unsere Struktur ein.

Der Returnwert von **stat** ist **0** bei Erfolg und irgendein anderer **int** bei Fehler (in diesem Fall: Fehlermeldung ausgeben und diesen File überspringen).

Doku zu **stat**:

<http://www.kernel.org/doc/man-pages/online/pages/man2/stat.2.html>

- In dieser **struct stat** interessieren uns 3 Member:
 - **st_size**, ein **int**, die *Größe* des Files in *Bytes*.
 - **st_mode** (auch ein **int**).
Dieses Feld enthält als Bitmuster u.a. den *Filetyp* und die Rechte.
Wir prüfen den Filetyp, indem wir vordefinierte Funktionen mit dem **st_mode**-Wert als Argument aufrufen:
Ist der File ein *normaler File*, liefert **S_ISREG** als Ergebnis **true**, und wir geben als Filetyp **'f'** aus. Sonst prüfen wir ihn mit **S_ISDIR**:
Es liefert **true**, wenn der Eintrag ein *Verzeichnis* ist.
In diesem Fall geben wir als Filetyp **'d'** aus. Liefert das auch **false** (z.B. weil der Eintrag ein Verweis ist), geben wir **'?'** als Filetyp aus.
 - **st_mtime** enthält das *Datum der letzten Änderung* (als **time_t**, d.h. Sekunden seit 1.1.1970). Das wandelt man am einfachsten mit der Funktion **ctime** (aus **time.h**) in eine Textdarstellung um.

Achtung:

- **ctime** möchte einen Pointer auf den **time_t**-Wert als Argument, nicht den Wert selbst!
- **ctime** liefert als Ergebnis einen **char**-Pointer auf ein internes statisches Array mit dem Datums- und Zeit-String.
Dummerweise enthält dieser String am Ende ein `'\n'`, das man vor der Ausgabe wegschneiden muss (außer man gibt das Filedatum als Letztes in der Zeile aus). Das geht am schnellsten, indem man das `'\n'` sucht (welche String-Funktion macht das?) und durch ein `'\0'` ersetzt.
- Wem das Ausgabe-Format von **ctime** nicht gefällt, der kann **st_mtime** mit einem **localtime**-Aufruf (wie in der vorigen Übung) in eine **struct tm** verwandeln und aus dieser **struct tm** die gewünschten Werte selbst mit einem **printf** schön formatiert ausgeben.

Schritt 3:

- In Version 3 machst du das **opendir** statt mit `."` mit dem auf der Befehlszeile angegebenen Verzeichnisnamen (wenn keiner da ist, nimmst du weiterhin `."`).
- Für den **stat**-Aufruf müssen wir uns jetzt den Filenamen des Files, zu dem wir die Fileinformation haben wollen, selbst zusammenbasteln, und zwar aus dem auf der Befehlszeile angegebenen Verzeichnisnamen, einem `"/` (dem Linux-Gegenstück zu `"\"` als Verzeichnis-Trennzeichen) und dem Filenamen im von **readdir** gelieferten Eintrag **d_name**.

Leg dir dafür ein **char-Array** an, kopiere die einzelnen Teile nacheinander mit **strcpy** und **strcat** hinein, und rufe dann **stat** mit diesem Namen auf. Prüfe zuerst, ob sie auch alle Platz haben (überspringe den Eintrag sonst).

Die maximale vom Betriebssystem unterstützte Länge von Filenamen incl. Verzeichnispfad ist durch die Konstante **PATH_MAX** aus dem Header **limits.h** gegeben, nimm das als Größe deines **char**-Arrays.

- 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...