Aufgabe 3: printdir (Abgabe bis Fr. 08.06.07 18:00 Uhr)

Entwickeln Sie ein Programm **printdir**, mit dem Sie — ähnlich wie mit dem UNIX-Kommando **ls(1)** — den Inhalt von Verzeichnissen (Katalogen / Dirtectories) anzeigen können. Die Namen der Verzeichnisse werden auf der Kommandozeile übergeben. Wird kein Verzeichnis übergeben, so soll Ihr Programm das aktuelle Verzeichnis (**"."**) ausgeben. Gehen Sie beim Entwurf des Programms in folgenden Arbeitschritten vor:

a) Geben Sie zunächst nur die Namen der Verzeichnisse aus, die Sie anzeigen sollen. Zeigen Sie hierzu jeden Verzeichnisnamen in einer eigenen Zeile gefolgt von einem Doppelpunkt an.

b) Schreiben Sie nun eine Funktion
`void printdir(const char *dirname),`
welche den Inhalt eines Verzeichnisses durch einen Tabulator eingerückt auf die Standardausgabe ausgibt. Diese Funktion soll für ein Verzeichnis jeweils nach der Ausgabe des Verzeichnisnamens aus Teilaufgabe a) aufgerufen werden. Der Name des Verzeichnisses wird hierbei als einziger Parameter übergeben. Wie beim UNIX-ls sollen dabei Dateien, deren Name mit einem **'.'** beginnt, nicht angezeigt werden, da es sich hierbei um versteckte Dateien handelt. Sollte ein Fehler beim Anzeigen eines Verzeichnisses auftreten, soll **printdir()** eine entsprechende Fehlermeldung liefern und mit der Ausgabe der übrigen Verzeichnisse fortfahren.

c) Erweitern Sie Ihr Programm nun so, dass die einzelnen Verzeichniseinträge sortiert ausgegeben werden. Verwenden Sie hierzu die Funktionen **qsort(3)** und **strcmp(3)**. Sie können in dieser Teilaufgabe zunächst davon ausgehen, dass ein Verzeichnis nicht mehr als 1000 Einträge enthält. Zum Sortieren mit der Funktion **qsort(3)** sollen Sie zunächst eine Datenstruktur aufbauen, deren Aufbau äquivalent zu der des **argv**-Arrays ist. Es soll also ein Array aus **char*** aufgebaut werden. Jeder dieser Zeiger zeigt auf den Anfang eines Datei- oder Verzeichnisnamens. Vergessen Sie nicht, dass die von **readdir(3)** zurückgelieferten Zeiger bei erneutem Aufruf von **readdir(3)** ihre Gültigkeit verlieren. Daher ist es notwendig, jeden Datei-/Verzeichnisnamen nach dem **readdir(3)**-Aufruf zu kopieren. Den hierfür notwendigen Speicher müssen Sie mit **malloc(3)** allokieren. Die Größe des Strings können Sie mit **strlen(3)** ermitteln, denken Sie aber an das zusätzlich notwendige Byte für das abschließende **'\0'**-Zeichen. Vergessen Sie nicht, allen von ihrem Programm allokierten Speicher auch wieder freizugeben.
Testen Sie Ihr Programm nun aber auch mit Verzeichnissen, die mehr als 1000 Einträge enthalten, wie z.B. dem Verzeichnis **/usr/bin**. Ihr Programm sollte sich im Idealfall mit einer Fehlermeldung beenden, keinesfalls aber abstürzen.

d) Ergänzen Sie ihr Programm nun so, dass es Verzeichnisse beliebiger Länge ausgeben kann. Hierzu müssen Sie das Array mit den **char*** nach Bedarf vergrössern. Verwenden Sie hierzu die Funktion **realloc(3).**

Hinweise: Sie finden im Verzeichnis **/proj/i4gdi/pub/aufgabe3** für jede Teilaufgabe eine Beispiellösung, deren Ausgabe Sie mit Ihrer eigenen Lösung vergleichen können. Für die Abgabe ist nur die endgültige Lösung notwendig, die Sie in einer Datei **printdir.c** in Ihrem Projektverzeichnis ablegen sollen.

Essentielle Funktionen: **opendir(3)**, **readdir(3), closedir(3), malloc(3), free(3), realloc(3), strlen(3), qsort(3)**