SPiC-Aufgabe #7: printdir

(12 Punkte, in Zweier-Gruppen)

Entwickeln Sie ein Programm `printdir`, das - ähnlich wie das UNIX-Kommando `ls(1)` - den Inhalt verschiedener Verzeichnisse ausgeben kann.

Es wird empfohlen beim Entwurf des Programms in folgenden Arbeitsschritten vorzugehen:

(a) Schreiben sie zunächst ein Programm, welches alle Einträge des aktuellen Verzeichnisses (`'.'`) als je einen Eintrag pro Zeile ausgibt. Einträge, deren Name mit einem `'.'` beginnen, sollen nicht angezeigt werden (versteckte Dateien). (`opendir(3)`, `readdir(3)`, `closedir(3)`)

(b) Erweitern Sie das Programm nun so, dass vor dem Dateinamen, auch die Dateigröße ausgegeben wird. Name und Größe sollen durch einen Tabulator (`'\t'`) getrennt werden. Am Ende der Ausgabe soll die Gesamtzahl der ausgegebenen Einträge, sowie deren Gesamtgröße ausgeben werden. Ignorieren Sie für die Ermittlung der Gesamtanzahl und -größe alle Einträge, bei denen es sich nicht um eine reguläre Datei handelt. (`lstat(2)`)

(c) Werten Sie nun die Parameter `argv` aus. Alle übergebenen Parameter sollen als Verzeichnispfade interpretiert werden und wie in (a) und (b) beschrieben ausgegeben werden. Die Ausgabe eines Verzeichnisses soll mit `'<Verzeichnisname>:\n'` beginnen. Wird kein Parameter übergeben, soll das aktuelle Verzeichnis ausgegeben werden.

Hinweise:

- Für die Abgabe ist nur die endgültige Lösung notwendig, die Sie in einer Datei `printdir.c` in Ihrem Projektverzeichnis ablegen sollen.
- Sie finden im Verzeichnis /proj/i4spic/pub/aufgabe7 eine Beispiellösung, deren Ausgabe Sie mit Ihrer eigenen Lösung vergleichen können.
- Ihr Programm muss nur Pfade und Dateinamen bis zu einer Gesamtlänge von 1024 Zeichen[1] behandeln können. Achten Sie bei zu langen Pfaden und Dateinamen auch hier auf eine entsprechende Fehlermeldung.
- Die Funktionen zur Behandlung von Zeichenketten aus `string.h` sind beim Lösen der Aufgabe hilfreich.
- Achten Sie auf aussagekräftige Fehlermeldungen, die alle auf dem Standardfehlerkanal ausgegeben werden sollen. (`fprintf(stderr,...)(3)` / `perror(3)`)
- Ihr Programm muss mit dem folgendem Aufruf übersetzen:
  `gcc -std=c11 -pedantic -D_XOPEN_SOURCE=700 -Wall -Werror -O3 -o printdir printdir.c`
  Diese Konfiguration wird zur Bewertung herangezogen.
- Testen Sie ihr Programm auch mit `valgrind`. Dies kann bei der Suche nach Fehlern helfen. *suppressed Errors* können ignoriert werden. Weitergehende Fehlermeldungen erhalten Sie, wenn Sie `valgrind` mit den Flags `--leak-check=full --show-reachable=yes` aufrufen.
- Alternativ können Sie `make` verwenden. Hierzu muss einmal
  `export CFLAGS="-std=c11 -pedantic -D_XOPEN_SOURCE=700 -Wall -Werror -O3"`
  angegeben werden. Anschließend können Sie Ihr Programm mit `make printdir` übersetzen.

Beispielausgabe

```
$ cd /proj/i4spic/pub/aufgabe7
$ ./printdir test/first_path test/second_path
test/first_path:
157     file2.txt
127     file1.txt
4096    test_dir
2 Dateien; 284 Bytes
test/second_path:
115     fileB.txt
4096    dir2
116     fileA.txt
4096    dir1
2 Dateien; 231 Bytes
```

Abgabezeitpunkt

T01	24.06.2018	18:00:00
T02	24.06.2018	18:00:00
T03	24.06.2018	18:00:00
T04	25.06.2018	18:00:00
T05	19.06.2018	18:00:00
T06	19.06.2018	18:00:00
T07	20.06.2018	18:00:00
T08	20.06.2018	18:00:00
T09	20.06.2018	18:00:00

[1] Alternativ kann auch `PATH_MAX` aus der `limits.h` verwendet werden.