Systemprogrammierung - AIN/2

Sommersemester 2024

Übungsaufgabe 4: C Aufzählungen, Strukturen, Varianten, Übersetzungseinheiten, POSIX-Funktionen

Abgabe bis 6./7.6.2024

Vorbereitung

Legen Sie ein Arbeitsverzeichnis für Aufgabe 4 an und speichern Sie darin das Archiv ↷ **aufgabe4.tar.gz**. Entpacken sie das Archiv mit dem folgenden Kommando:

```
tar xzf aufgabe4.tar.gz
```

Ihr Arbeitsverzeichnis von Aufgabe 4 sollte anschließend folgende Dateien enthalten: `Makefile`, `listfiles.c`, `listfiles-out.txt`, `listfiles-example-dir`

Programmierung

Das vorgegebene Programm `listfiles.c` listet Verzeichnisinhalte rekursiv auf. In `listfiles-out.txt` finden Sie eine Beispielausgabe, die so ähnlich auch das Linux-Kommando `ls -R1` liefern würde. Lesen Sie den Programmtext durch und erstellen Sie dann die darin verwendete Übersetzungseinheit `fileinfo` mit der folgenden Schnittstelle:

- einem `struct`-Typ `fileinfo` mit eingebetteter anonymer `union` und zugeordnetem `enum`-Typ `filetype` (siehe Vorlage in den Vorlesungsunterlagen).

  Die Struktur soll eine Listenverkettung in Form eines Zeigers auf `struct fileinfo` und ein Array von Zeichen für den Dateinamen enthalten. Verwenden Sie als Array-Größe die symbolische Konstante `NAME_MAX` aus dem POSIX-Header `limits.h` plus `1`. Mit der zugeordneten `enum` unterscheiden Sie zwischen den Dateitypen `filetype_regular`, `filetype_directory` und `filetype_other`. Mit der eingebetteten `union` speichern Sie bei regulären Dateien die Dateigröße in Byte, bei Verzeichnissen eine `fileinfo`-Liste in Form eines Zeigers auf das erste Listenelement.

  Deklarieren Sie einen Aliasnamen `fileinfo` für Ihren `struct`-Typ.

- einer Funktion `fileinfo_create`.

  Die Funktion soll einen Eingabeparameter für einen Dateinamen haben und als Rückgabewert einen Zeiger auf eine `fileinfo`-Struktur liefern, die mit den Werten zu der benannten Datei initialisiert ist. Im Fehlerfall soll die Funktion `NULL` zurückgeben. Ist der Fehler ein zu langer Dateinamen, soll außerdem die globale POSIX-Variable `errno` auf `ENAMETOOLONG` gesetzt werden.

  Verwenden Sie die POSIX-Funktion `lstat`, um die Art der Datei und im Falle einer regulären Datei deren Größe in Byte festzustellen (siehe `man 2 lstat` oder die POSIX-Dokumentation im Internet).

  Im Falle eines Verzeichnisses müssen Sie eine Liste von `fileinfo`-Strukturen für alle Dateien im Verzeichnis erstellen. Schreiben Sie dafür eine private Hilfsfunktion `list_directory`, die das

Verzeichnis mit Hilfe der POSIX-Funktionen `opendir`, `readdir` und `closedir` liest (siehe Vorlesungsbeispiel). Die Funktion muss das aktuelle Arbeitsverzeichis mit der POSIX-Funktion `chdir` auf den Namen des bearbeiteten Verzeichnisses setzen und vor dem `return` mit `".."` wieder zurücksetzen (siehe `man 2 chdir` oder die POSIX-Dokumentation im Internet). Für das Aufbauen der Liste können Sie sich an der vorgegebenen `main`-Funktion orientieren. Hier entsteht Rekursion, weil die aus `fileinfo_create` aufgerufene Funktion `list_directory` wiederum `fileinfo_create` aufruft. Achten Sie darauf, die Verzeichniseinträge mit Namen `"."` und `".."` zu überspringen, um eine endlose Rekursion zu verhindern.

- einer Funktion `fileinfo_print`.

  Die Funktion soll einen Eingabeparameter für einen Zeiger auf eine `fileinfo`-Struktur haben und keinen Rückgabewert. Sie soll je nach Dateityp eine von drei privaten Hilfsfunktionen `print_regular`, `print_directory` oder `print_other` mit den jeweiligen Werten aus der `fileinfo`-Struktur aufrufen.

  `print_regular` soll zwei Eingabeparameter haben, einen für den Dateinamen und einen für die Dateigröße. Die beiden Werte sollen auf die Standardausgabe geschrieben werden. Das Format können Sie der Beispielausgabe in `listfiles-out.txt` entnehmen. Die Beispielausgabe ist mit dem Aufruf
  `./listfiles xxx listfiles.c listfiles-example-dir /dev/tty > listfiles-out.txt 2>&1`
  der Musterlösung erzeugt.

  `print_directory` soll drei Eingabeparameter haben, einen für den Pfad bis zum auszugebenden Verzeichnis (beim Aufruf in `fileinfo_print` mit `""` belegen), einen für den Dateinamen des auszugebenden Verzeichnisses und einen für die Liste der Unterverzeichnisse in Form eines `fileinfo`-Zeigers. Zum Ausgabeformat siehe wieder `listfiles-out.txt`. Rufen Sie für jedes Unterverzeichnis wiederum `print_directory` auf.

  `print_other` soll einen Eingabeparameter für den Dateinamen haben. Zum Ausgabeformat siehe auch hier `listfiles-out.txt`.

- einer Funktion `fileinfo_destroy`.

  Die Funktion soll einen Eingabeparameter für einen Zeiger auf eine `fileinfo`-Struktur haben und keinen Rückgabewert. Sie soll den Speicher der übergebenen Struktur und im Falle des Dateityps Verzeichnis auch der Strukturen aller Unterverzeichnisse freigeben.

Test und Qualitätssicherung

Verwenden Sie zum Testen die folgenden Befehle und probieren Sie verschiedene Eingaben, insbesondere auch `hallo` als einzige Eingabe:

```
make
make cppcheck
valgrind ./listfiles *
```

Führen Sie auch den folgenden automatisierten Test aus:

```
./listfiles xxx listfiles.c listfiles-example-dir /dev/tty > out.txt 2>&1
diff -Z listfiles-out.txt out.txt
```

- `valgrind` darf keine Fehler und `diff` keine Unterschiede melden.
- `cppcheck` sollte keine Probleme melden.

Bessern Sie gegebenenfalls nach.

Abgabe

Führen Sie Ihr Programm mit den automatisierten Tests vor.
Zeigen Sie das ausgefüllte Teilnahmeprotokoll.

Hinweis:
Der Compiler `gcc` darf für Ihr Programm keine Fehler oder Warnungen mehr ausgeben.
Ihr Programm muss außerdem ordentlich formatierte sein. Bessern Sie die Formatierung gegebenenfalls mit `astyle` nach:
`astyle -p -H --style=ansi *.[ch]`

Freiwillige Zusatzaufgabe *(1 Bonuspunkt)*

Erstellen Sie ein C-Programm `filesize`, das als Kommandozeilenargumente beliebig viele Dateinamen erwartet und für jede der Dateien die Größe in Byte auf die Standardausgabe schreibt.

Bei einem Aufruf ohne Kommandozeilenargumente soll das Programm mit der POSIX-Funktion `read` byteweise von der Standardeingabe lesen und dabei die Anzahl der Bytes zählen. Andernfalls soll das Programm für die angegebenen Dateien zur Größenbestimmung die POSIX-Funktion `stat` verwenden. Siehe `man 2 read` und `man 2 stat` oder die POSIX-Dokumentation im Internet. Achten Sie auf eine korrekte Fehlerbehandlung mit Ausgabe der zugehörigen Systemmeldung. Damit die Systemmeldung in der auf dem Rechner (bzw. im beim Aufruf verwendeten Terminal) eingestellten Sprache erscheint, müssen Sie am Anfang des Programms folgenden Aufruf einfügen (siehe auch `man 3 setlocale`):

```
setlocale(LC_ALL, "");
```

Hinweise:
Verwenden Sie das Vorlesungsbeispiel `count.c` als Vorlage. Ersetzen Sie im Fall der Standardeingabe den C-Bibliotheksaufruf `fgetc` durch den Aufruf der POSIX-Funktion `read` und bei Dateien die ganze Lese-/Zähl-Schleife durch den Aufruf der POSIX-Funktion `stat`. Den C-Bibliotheksaufruf `fopen` brauchen Sie nicht mehr. Beachten Sie außerdem, dass POSIX für ganzahlige Datentypen in der Regel Aliasnamen verwendet. Ihr Programm muss damit richtig umgehen.

Prof. Dr. H. Drachenfels
Hochschule Konstanz** - **Impressum** - **Datenschutzerklärung

Letzte Änderung: 28.3.2024