Blatt 12

Abgabe bis: 02. Februar 2012

Organisatorisches

Dies ist das letzte bewertete Uebungsblatt. Wenn ihr dieses abgeschlossen habt, konntet ihr auf 12 Blaettern, in 15 Aufgaben, insgesamt 130 Punkte sammeln. **Zum Bestehen der Vorleistung sind 65 Punkte (50%) noetig.**

Meldet euch baldmoeglichst im Portal fuer die Vorleistung an. Wir bestaetigen dann euer Bestehen, falls ihr die 65 Punkte erreicht habt. Danach koennt ihr euch fuer die Klausur am Montag, 20. Februar anmelden.

Zur Pruefungsvorbereitung findet ihr auf der Vorlesungswebsite eine Probeklausur.

In den verbleibenden Wochen bis zur Klausur werden weiterhin Uebungen stattfinden.

15. dircmp (10P)

Schreibt das Programm `dircmp`, das zwei Verzeichnisbaeume vergleicht und ausgibt, ob Dateien nur im einen, nur im anderen, oder in beiden Verzeichnisbaeumen vorhanden sind.

Die Datenstrukturen und das Interface sind in den Header-Dateien *dirtree.h*, *path.h*, und *scandir.h* schon vorgegeben. Nehmt sie als Ausgangsbasis und implementiert die einzelnen Funktionen. (Wer will, darf auch eine komplett eigene Loesung entwickeln.) Das Programm ist in Einzelmodule unterteilt, die unabhaengig voneinander implementiert werden koennen. Dadurch ist es einfach, die Programmierung im Team aufzuteilen.

`path.c`
: enthaelt nur die Stringverkettungsfunktion `make_path()`. Das sollte kein Problem sein.

`scandir.c`
: enthaelt die Funktion `scan_directory()`, die fuer jedes der beiden Wurzelverzeichnisse einmal aufgerufen werden muss. Sie durchlaeuft die Baeume rekursiv und baut die Datenstrukturen auf.

`dirtree.c`
: enthaelt Funktionen zur Verwaltung der verwendeten Datenstrukturen.

`dircmp.c`
: das Hauptprogramm.

Ihr muesst fuer beide Verzeichnisbaeume beim Durchlauf mit *scan_directory()* jeweils eine gleichartige dynamische Datenstruktur aufbauen. Diese besteht je Verzeichnis aus einer `DirNode` mit `DirEntry`s fuer jeden Verzeichniseintrag. Die `DirEntry`s haengen als binaerer Suchbaum an der `DirNode`. Zusaetzlich sind alle `DirEntry`s eines Verzeichnisses zu einer sortierten doppelt verketteten Liste verkettet. Man kann die Eintraege eines Verzeichnisses also baumartig oder linear durchlaufen. Ist ein Eintrag ein Verzeichnis, dann haengt an ihm wiederum eine komplette solche Verzeichnisdatenstruktur, bestehend aus einer `DirNode` und mehr oder weniger vielen `DirEntry`s.

Mithilfe der Iteratoren, die ueber die verketteten Listen wandern, koennen beide Verzeichnisbaeume in einem Durchlauf miteinander verglichen werden.

Am Ende soll euer Programm die Vereinigung aller Pfadnamen ausgeben. Bei jedem Eintrag soll gekennzeichnet sein, in welchen der drei Faelle er gehoert:

(1) Nur in Verzeichnisbaum 1 (Prefix '`1 :`')

(2) Nur in Verzeichnisbaum 2 (Prefix '` 2:`')

(3) In beiden Verzeichnisbaeumen (Prefix '`12:`')

Auf der Vorlesungswebsite findet ihr fertige Objektdateien (fuer Solaris und GNU/Linux). Diese koennt ihr zum Testen verwenden.

Beispielausgabe:

```
$ ./dircmp dir1 dir2
12: .a
12: b
12: c
12: c/d
 2: c/f
12: c/g
1 : c/h
12: i
12: i/j
1 : i/k
 2: i/m
12: n
$
```

Zum Einreichen eurer Loesung:

```
submit ss1 15 team [notes] makefile dircmp.c
dirtree.h dirtree.c path.c path.h scandir.c scandir.h
```

Fragen zur Selbstkontrolle

- Erklaere das Konzept von *setuid* bzw. *setgid*. Nenne ein sinnvolles Einsatzbeispiel.
- Wer hat dieses Konzept erfunden und patentiert?

- Nenne ein paar typische Schwachstellen in C.
- Warum machen Strings bei C mehr Probleme als in vielen anderen Programmiersprachen? Warum hat man sie in C nicht anders implementiert?
- Wie hilft die *libowfat* dabei? Erklaere das `struct stralloc`.
- Nenne ein paar stralloc-Funktionen. Zeige ein einfaches Beispiel.
- Wann sind Listen, Hashes, Baeume geeignet? Wann nicht?