ulm university universität
uulm

Institut für Numerische Mathematik

Dr. Andreas F. Borchert
Moritz Carmesin

10. Januar 2020
Blatt 9

Systemnahe Software I (WS 2019/2020)

Abgabe bis zum 17. Januar 2020, 14:00 Uhr

Lernziele:

- Umgang mit der *opendir*-Schnittstelle zur rekursiven Traverse durch Verzeichnishierarchien
- Auslesen von Inode-Informationen mit Hilfe der Systemaufrufe *stat* bzw. *lstat*

Aufgabe 11: Neueste Dateien finden

Das Ziel dieser Aufgabe ist es, ein Programm zu schreiben, das die neuesten Dateien findet und anzeigt.

Per Voreinstellung sollte Ihr Programm die Verzeichnishierarchie beginnend ab dem aktuellen Verzeichnis (".") traversieren, es sei denn, es wird explizit als letztes Argument der Name eines Verzeichnisses angegeben. Die folgenden Optionen sind ebenfalls zu unterstützen:

- „-c“: Anzahl der Dateien, die angezeigt werden sollen.
- „-d“: Legt fest, ob das Datum der letzten Änderung angezeigt werden soll.
- „-o“: Falls gesetzt, werden die ältesten Dateien angezeigt.

Bei fehlerhaften Angaben auf der Kommandozeile ist eine „Usage“-Meldung auszugeben.

Hier ist ein beispielhafter Aufruf, bei der die zuletzt ausgegebene Datei die neueste ist:

```
theon$ findnewest -c 5
./Makefile
./traverse.o
./findnewest.o
./heap.o
./findnewest
theon$
```

Hinweise

Wie in den Unterlagen zur Vorlesung beschrieben, kann der Inhalt eines Verzeichnisses mit den Funktionen *opendir*, *readdir* und *closedir* ausgelesen werden. Um symbolischen Verweisen nicht ungewollt nachzufolgen, sollten die Verwaltungsinformationen einer Datei mit dem Systemaufruf *lstat* an Stelle von *stat* abgerufen werden. Hierzu empfiehlt sich ein Blick auf die entsprechenden Manualseiten im POSIX-Standard: *lstat* und *stat.h*.

Ebenso ist auch darauf zu achten, dass *readdir* auch die Einträge für die Verzeichnisse „." und „.." zurückgibt. Diese sind zu ignorieren. (Warum?)

Um Verzeichnisse und symbolische Verweise als solche zu erkennen, empfiehlt sich die Verwendung der in <*sys/stat.h*> definierten Makros *S_ISDIR* und *S_ISLNK*, die jeweils das *st_mode*-Feld der von *lstat* gefüllten **struct** *stat* erhalten.

In der **struct** *stat* finden Sie in *st_mtime* sekundengenau den Zeitstempel der letzten Veränderung. Entsprechend dem neueren POSIX-Standard steht unter *st_mtim* (ohne abschließendes „e") eine **struct** *timespec* zur Verfügung, die über die Komponenten *tv_sec* (Sekunden) und *tv_nsec* (Nanosekunden) eine sehr viel genauere Zeit liefert (mit der Auflösung der lokalen Uhr). Im Rahmen der Aufgabe sollte die genauere Zeitangabe genutzt werden.

Beachten Sie, dass die Pfadnamen innerhalb einer tieferen Hierarchie beliebig lange werden können und daher dynamisch belegt und wieder freigegeben werden müssen. Ebenso ist darauf zu achten, die geöffneten Verzeichnisse wieder mit *closedir* zu schließen sind.

Wenn Sie die *n* neuesten (bzw. ältesten bei der Option „-o") Dateien ermitteln möchten, ist es sinnvoll, die Datenstruktur eines Heaps mit bis zu *n* Elementen zu verwenden. Bei einem Heap hat das Ermitteln des ältesten bzw. jüngsten Elements konstanten Aufwand. Wenn Sie sehen, dass die aktuell betrachtete Datei jünger ist als die älteste der *n* bislang neuesten Dateien, dann kann dieses Element im Heap ausgetauscht werden mit logarithmischen Aufwand. Ebenso unterstützt der Heap am Schluss die Extraktion der *n* neuesten Elemente in sortierter Reihenfolge (beginnend mit der ältesten), ebenfalls jeweils mit logarithmischen Aufwand. Hinweise zu dieser auf einem Array der Länge *n* basierenden Datenstruktur finden sich unter diesen Webseiten: `https://www.cs.cmu.edu/~adamchik/15-121/lectures/Binary%20Heaps/heaps.html`,`https://en.wikipedia.org/wiki/Heap_(data_structure)`.

Wie üblich ist dies wieder eine Aufgabe, die sich hervorragend dafür eignet, innerhalb einer Gruppe aufgeteilt zu werden:

- Die Datenstruktur für den Heap zusammen mit den zugehörigen Operationen sollte als separates Modul realisiert werden.
- Eine rekursive Traverse durch das Dateisystem lässt sich als getrennte Funktion realisieren – ganz unabhängig von dem Einsatzzweck. Dies ist möglich, indem die Traverse einen per Funktionszeiger übergebene Funktion für jede der gefundenen Dateien aufruft. Da diese Datei ohnehin von der Traverse mit *lstat* untersucht werden muss, kann dieser Funktion der Zeiger auf den **struct** *stat* gleich mit übergeben werden. Typischerweise benötigt die so aufgerufene Funktion einen Kontext (wie hier zum Beispiel einen Zeiger auf den Heap). Da dieser außerhalb des Traverse-Moduls definiert wird, wird hierfür am besten ein **void**∗ verwendet. So könnte das aussehen:

  typedef *bool* (∗*TraverseHandler*)(**void**∗ *handle*, **char**∗ *path*, **struct** *stat*∗ *statbuf*);

bool traverse(**char**∗ *path*, **void**∗ *handle*, *TraverseHandler handler*);

- Im Hauptprogramm sollten Sie die Argumente aus der Kommandozeile verarbeiten, den Heap anlegen, die Traverse starten, wobei sie eine Funktion des Hauptprogramms als Parameter übergeben. Diese Funktion sollte dann den Heap entsprechend befüllen. Nach dem Abschluss der Traverse werden schließlich die Elemente des Heaps ausgegeben.

Reichen Sie bitte Ihre Lösung mit folgendem Kommando ein:

```
theon$ submit ss1 11 findnewest.c \
   [heap.c heap.h] \
   [traverse.c traverse.h]
```

Viel Erfolg!