Grundlagenpraktikum Rechnerarchitektur (GRA)

Tutorübung

Moritz Beckel

23.05.2022 16:00 / 27.05.2022 15:00

Organisatorisches

Zusatzaufgabe 8: Union Find

- Falls 8 ETCS benötigt werden
- Abgabe bis 2022-06-12 23:59 (3 Wochen)
- Muss erfolgreich bearbeitet werden zum bestehen des Praktikums

Hausaufgaben

- Memccpy
- Map
- Quiz

T4.1 Parsen mit getopt

Vorlage: `https://gra.caps.in.tum.de/m/numquad.tar`

1. Bevor Sie mit der Implementierung des CLI beginnen, informieren Sie sich zunächst über die Funktion `getopt`. Nutzen Sie den Befehl `whatis getopt` (und evtl. `man man`) um herauszufinden, welche `man` page die relevanten Informationen der C-Funktion `getopt` beinhaltet. Lassen Sie sich diese anzeigen.

2. Überschaffen Sie sich zunächst in den Abschnitten `NAME` und `SYNOPSIS` einen Überblick über die bereitgestellte Funktionalität und Signatur der `getopt` Funktion. Lesen Sie auch den ersten Absatz des Abschnitts `DESCRIPTION` um ein grobes Verständnis der Funktionsweise von `getopt` zu erlangen.

3. Welchen Header/Welche Header müssen Sie einbinden, um `getopt` verwenden zu können? Öffnen Sie die Datei `main.c` und fügen Sie diesen hinzu.

T4.1 Parsen mit getopt

4. Die `getopt` Parameter `argc` und `argv` entsprechen den bereits bekannten Parametern der `main` Funktion. Finden Sie die Bedeutung des `optstring` Parameters heraus.
   Hinweis: Mit `/optstring<ENTER>` können Sie in der `man` page nach dem Wort „optstring“ suchen. Weitere Suchergebnisse können Sie mit `n` (next) anzeigen lassen.

5. Betrachten Sie die Usage- und Help-Messages am Anfang der Datei `main.c`. Formulieren Sie einen `optstring`, der das Parsen dieses CLI ermöglicht.

6. Welche Rückgabewerte hat die Funktion `getopt` für den eben formulierten `optstring`? Betrachten Sie hierzu den Abschnitt `RETURN VALUE` der `man` page.

T4.1 Parsen mit getopt

7. Integrieren Sie nun das Parsen der Optionen `-t` und `-h` in das Programm. Für andere Optionen soll die Programmausführung nach Ausgabe der Usage-Meldung zunächst abgebrochen werden.

8. Die Optionen `-a`, `-b` und `-n` erwarten jeweils ein Argument. Finden Sie mithilfe des Abschnitts `DESCRIPTION` und/oder `EXAMPLE` der `man` page heraus, wie `getopt` diese beim Parsen zur Verfügung stellt.

T4.1 Parsen mit getopt

9. Integrieren Sie nun auch das Parsen der verbleibenden Optionen. Für unsere Zwecke ist es ausreichend, `a` und `b` mit `atof` und `n` mit `atol`[1] zu konvertieren.

10. Zuletzt wollen wir nun noch das Programmargument `fn` parsen. Finden Sie auch hierfür zunächst mithilfe der `man` page heraus, wie Sie darauf zugreifen können. Implementieren Sie basierend darauf die Parse-Funktionalität.

Hausaufgaben

P5.1 Kreisumfang

Aufgabe: Schreiben Sie eine Funktion, die den Umfang eines Kreises mit einem gegebenen Radius r berechnet: $circ(r) = 2\pi r$

```
double circ(double radius);
```

Hausaufgaben

P5.2 Numerische Quadratur [4 Pkt.]

Für eine gegebene Funktion $f : \mathbb{R} \to \mathbb{R}$ soll auf dem Intervall $[a; b]$ das Integral approximiert werden. f soll in dem Intervall an n gleichmäßig verteilten Stützstellen evaluiert werden. Zwischen den Stützstellen soll linear interpoliert werden. Im Fall $n = 2$ wird die Funktion also genau an den Stellen a und b evaluiert und die Fläche des Trapezes ist:

$$\frac{(b-a) \cdot (f(b) + f(a))}{2}$$

Bei $n = 3$ sind die Stützstellen folglich $\left\{a, \frac{a+b}{2}, b\right\}$ und es wird die Fläche von zwei Trapezen berechnet. Für $n < 2$ ist das Ergebnis undefiniert, dies soll in der Implementierung durch den Wert *NaN* dargestellt werden.

Aufgabe: Implementieren Sie folgende Funktion in Assembler:

```
double numquad(double(* f)(double), double a, double b, size_t n);
```

Hausaufgaben

P5.3 Round [2 Pkt.]

In dieser Aufgabe soll in C eine Funktion implementiert werden, welche eine rationale Zahl zur einer ganzen Zahl rundet. Die Eingabezahl ist dabei entweder eine Double-Precision Floating-Point Zahl oder eine 32.32-Bit Fixed-Point Zahl; gerundet soll entweder zur nächst größeren oder zur nächst kleineren ganzen Zahl. Das Ergebnis soll in gleiche Format wie die Eingabe zurückgegeben werden.

Eine Zahl wird durch folgende Datenstruktur dargestellt. Ob eine Zahl als Floating-Point-Zahl dargestellt wird, wird durch `isflt` angegeben. Innerhalb der `union`[3] wird die Fixed-Point-Zahl als `int64_t` oder die Floating-Point-Zahl als `double` gespeichert.

```
struct num { bool isflt; union { int64_t fix; double flt; }; };
```

Die Funktion hat folgende Signatur:

```
enum RoundMode { RM_FLOOR = 1, RM_CEIL };
struct num num_round(struct num num, enum RoundMode rm);
```