OPERATING SYSTEMS BEISPIEL 1

Aufgabenstellung A – postfixcalc

Schreiben Sie einen Rechner, der Ausdrücke in Postfix-Notation interprätiert und auswertet. Implementieren Sie die vier Grundrechenarten ($+$, $-$, $*$, $/$) sowie die Winkelfunktionen Sinus und Cosinus. Die Option `-i` erzwingt eine ganzzahlige Ausgabe. Wird `-a` angegeben, ist nach der Berechnung der Betrag (Absolutwert) auszugeben. Die maximale Zeilenlänge ist auf 1024 Zeichen beschränkt.

```
SYNOPSIS
    calc [-i] [-a] [file1 [file2 ...]]
```

Postfix ist eine Notation, die ohne Klammerung auskommt und daher für Maschinen leicht lesbar ist. Im Gegensatz zur Infix-Notation wird die Operation immer nach den Operanden angegeben. Dazu einige Beispiele:

Infix	Postfix
$1 + 2$	$1\ 2\ +$
$2 * (1 + 3)$	$2\ 1\ 3\ +\ *$
$\sin(1/5)$	$1\ 5\ /\ \sin$
$(1 + 2) * (3 + 4)$	$1\ 2\ +\ 3\ 4\ +\ *$

Anleitung

Lesen Sie aus der Datei bzw. den Dateien zeilenweise bis zum Erreichen von *EOF*. Wenn keine Datei angegeben ist, soll von *stdin* gelesen werden. Da jede Zeile einer Rechnung entsprechen soll, können Sie die Berechnung sofort ausführen.

Verwenden Sie einen Stack um die Ergebnisse zu berechnen. Extrahieren Sie wiederholt mit *strtod(3)* eine Zahl aus der Zeile und legen Sie diese oben auf den Stack. Konnte keine Zahl konvertiert werden, überprüfen Sie, ob ein korrekter Operator angegeben wurde. In diesen Fall nehmen Sie zwei Zahlen (für die binären Operatoren $+$, -, $/$ und *) bzw. eine Zahl (für die unären Operatoren s und c) vom Stack und führen die entsprechende Berechnung durch. Das Ergebnis soll dann wiederum auf den Stack gelegt werden. Wenn das Zeilenende erreicht wurde, darf sich nur noch ein Element – das Ergebnis – am Stack befinden.

Stack

Wir empfehlen für dieses Beispiel eine simple, statische Stackimplementierung. Sie können sich beispielsweise an diesem Codesnippet orientieren:

```
#define STACK_SIZE 1024

static int stack[STACK_SIZE];
static int stack_top;

void stack_push(int a)
{
  stack[stack_top] = a;
  ++stack_top;
}

int stack_pop(void)
{
```

```
  --stack_top;
  return stack[stack_top];
}
```

Versuchen Sie eine möglichst knappe maximale Stack Größe `STACK_SIZE` zu wählen.

Testen

Testen Sie Ihr Programm mit verschieden Eingaben. Erstellen sie zum Beispiel eine Testdatei `t1` mit folgenden Zeilen:

```
2 3 +
2 3 * 4 -
3 2 s* -4.5 +
1.3E-2 100 *
```

Rufen Sie Ihr Programm dann wie folgt auf:

```
$ ./calc t1
  5.000000
  2.000000
 -1.772108
  1.300000

$ ./calc < t1
  5.000000
  2.000000
 -1.772108
  1.300000

$ ./calc -i -a t1
  5
  2
  1
  1
```

Hinweis

Leerzeichen zwischen Operatoren und Zahlen sind erlaubt, aber nicht notwendig. `1 2 -3 +` ist nicht gültig, da das Minus zur `3` gehört und von *strtod(3)* konsumiert wird. `1 2 *3 +` oder `1 2 - 3 +` hingegen ist in Ordnung. Für die Winkelfunktionen müssen Sie `math.h` einbinden, sowie beim Linken den Parameter `-lm` verwenden.

Richtlinien

Bitte beachten Sie auch die *Richtlinien für die Erstellung von C-Programmen* auf der Übungswebsite.