Lehrstuhl für Connected Mobility
TUM School of Computation, Information and Technology
Technische Universität München

Grundlagen - Betriebssysteme und Systemsoftware

IN0009, WiSe 2023/24

Übungsblatt 1

23. Oktober 2023 – 27. Oktober 2023

Hinweis: Mit * gekennzeichnete Teilaufgaben sind ohne Lösung vorhergehender Teilaufgaben lösbar.

Aufgabe 1 C the difference

Vorbereitung: Vorbereitend auf diese Aufgabe sollten Sie den C-Primer[1] von Jonas Pfoh lesen.

Die Programmiersprache C verhält sich in vielen Aspekten anders als die Ihnen bekannte Sprache Java. Ihr Tutor wird Ihnen anhand des folgenden Beispielprogramms zur Berechnung der Fakultät einige Grundlagen und Besonderheiten von C erläutern.

```java
import java.util.Scanner;

public class Fakultaet
{
  public static void main (String[] args)
  {
    Scanner scan = new Scanner(System.in);

    int fakultaet = fak(scan.nextInt());
    System.out.println("Fakultaet: " + fakultaet);
  }
  private static int fak(int x) {
      return x <= 1 ? 1 : (x*fak(x-1));
  }
}
```

```c
#include <stdlib.h>
#include <stdio.h>

int fak(int);

int main(int argc, char *argv[]) {
  char *buf = malloc(100 * sizeof(char));
  fgets(buf, 100, stdin);

  int fakultaet = fak(atoi(buf));
  printf("Fakultaet: %d\n", fakultaet);
  free(buf);
}

int fak(int x) {
  return x <= 1 ? 1 : (x*fak(x-1));
}
```

- Java hat *import ...* und C *#include ...*.
- Java benötigt die *main(...)* in einer Klasse.
- Java hat einen automatisierten Garbage Collector.
- C hat manuelles Memory Management (*malloc(...)* und *free(...)*).
- In C muss beachtet werden, dass jeder String mit einem *Nullbyte* terminiert werden sollte. *fgets(...)* weiß dies und liest deswegen nur maximal 99 Zeichen und fügt am Ende das *Nullbyte* hinzu.
- In C muss die Methodensignatur von *fak(...)* vor der Benutzung deklariert werden.
- **Zusätzlich:** Durch den rekursiven aufruf von *fak(...)* kann es bei großen Zahlen zu einem Stackoverflow kommen.

Aufgabe 2 Binär- und Dezimalpräfixe

Einheitenpräfixe dienen dazu, Basiseinheiten zu skalieren. Ein bekanntes Beispiel hierfür ist *k* (Kilo) in km oder kg. Im Alltag werden häufig Dezimalpräfixe verwendet, die auf Potenzen der Zahl 10 basieren.

Im Kontext von Betriebssystemen werden jedoch häufig Binärpräfixe (Potenzen von 2) verwendet.

[1] https://gbs.cm.in.tum.de/media/material/c-primer.pdf

Binär			Dezimal
Kibi	$1024 = 2^{10}$	$1000 = 10^3$	Kilo
Mebi	2^{20}	10^6	Mega
Gibi	2^{30}	10^9	Giga
Tebi	2^{40}	10^{12}	Tera

Um diese zu kennzeichnen, wird zwischen dem Präfix und der Einheit noch ein *i* eingefügt. Aus kB wird also KiB (Kibibyte), aus MB wird MiB (Mebibyte).

a)* Übersetzen Sie von Binär- zu Dezimalpräfix: 2 KiB, 3 MiB, 4 GiB.

2 KiB = $2 * 1024$ B = 2048 B = $2,048$ kB
3 MiB = $3 * 1024^2$ B = 3145728 B $\approx 3,146$ MB
4 GiB = $4 * 1024^3$ B = 4294967296 B $\approx 4,295$ GB

b)* Übersetzen Sie von Dezimal- zu Binärpräfix: 4 kB, 3 MB, 2 GB.

4 kB = $4 * 1000$ B $\approx 3,906$ KiB
3 MB = $3 * 1000^2$ B = 3000000 B $\approx 2929,688$ KiB $\approx 2,861$ MiB
2 GB = $2 * 1000^3$ B = 2000000000 B $\approx 1,863$ GiB

c)* Hersteller von Speichermedien preisen diese mit Kapazitäten an, die auf Dezimalpräfixen basieren. Das führt häufig zu Verwirrung, da Software meist Binärpräfixe verwendet, jedoch die falschen Einheiten anzeigt (z.B. GB statt GiB). Wie viel Speicherplatz wird dem Käufer einer externen 2 TB-Festplatte nach dem Anschließen an ein solches System angezeigt?

1,82 TB, ihm „fehlen“ also knapp 10%.

Aufgabe 3 Bin verHext

Im Verlaufe dieser Veranstaltung werden Sie mit Werten in unterschiedlichen Basen umgehen müssen, sowie diese von einer in die andere Basis überführen. In GBS sind besonders die Basen 2, 10 und 16 relevant. Das Umrechnen von Zahlen zwischen diesen Basen sollten Sie unbedingt beherrschen.

Übersetzen Sie die gegebenen Werte von einer Basis in die Andere.

a)* Binär → Hex

0b10 1010 0x2A
0b1 1100 0111 0x1C7
0b1100 0000 1101 1110 0xC0DE

b)* Hex → Binär

0xAFFE 0b1010 1111 1111 1110
0xBADE 0b1011 1010 1101 1110
0xC0FFEE 0b1100 0000 1111 1111 1110 1110

c)* Dezimal → Hex

123 0x7B
65 0x41
262 0x106

d)* Dezimal → Binär

255 0b1111 1111
99 0b110 0011
54 0b11 0110

e)* Hex → Dezimal

0xABC 2748
0x64 100
0x420 1056

Noch Fragen? Dann schaut doch mal im Zulip-Stream vorbei:
`https://zulip.in.tum.de`

Aufgabe 4 Pointerarithmetik und Operatorpräzedenz

Die folgenden Deklarationen bilden die Grundlage für alle Teilaufgaben dieser Aufgabe. Verstehen Sie zunächst, welche Variablen deklariert und initialisiert werden, und welche Typen diese besitzen.

```
1  int arrayXYZ[10] = {0};
2  int i = 0, intVar = 0;
3  int *pi = 0;
4  for (i = 0; i < 10; i++)
5      arrayXYZ[i] = i;
```

a)* Was ist der Inhalt der Variable `pi` jeweils nach den folgenden Statements?

```
1  pi = &arrayXYZ[7];
2  pi = arrayXYZ;
3  pi = &arrayXYZ[0];
```

Dem Zeiger `pi` wird eine Referenz auf die Variable `arrayXYZ[7]` zugewiesen. `pi` zeigt auf die Adresse von `arrayXYZ[7]`. Zu beachten ist, dass nach Operatorpräzedenz in C `[]` stärker bindet als `&` (und `*`). Danach wird `pi` zweimal eine Referenz auf das erste Element des Arrays zugewiesen.

b)* Sind die folgenden Zuweisungen äquivalent? Verdeutlichen Sie sich, wie ein Array-Zugriff in C umgesetzt wird.

```
1  intVar = arrayXYZ[8];
2  intVar = *(arrayXYZ+8);
3  intVar = *(int *) ((void *) arrayXYZ+(8*sizeof(int)));
```

Es gibt keinen Unterschied zwischen den drei Anweisungen. Alle greifen auf das neunte Element des Arrays zu. Der Compiler kennt die Größe eines Arrayelements zur Compile-Time, daher sind die zweite und dritte äquivalent.

c)* Kompilieren die folgenden Statements ohne Warnings oder Errors? Wenn nicht, was ist das Problem? Können Fehler zur Laufzeit auftreten? Wie würden sich diese Fehler zur Laufzeit auswirken?

```
1    i = pi;
2    intVar = i;
3    *(arrayXYZ+3) = 3;
4    arrayXYZ[1320] = "a";
5    arrayXYZ[1] = &*(arrayXYZ + 15);
```

Statements 2 und 3 kompilieren ohne Fehler. Statement 1 und 5 ergeben eine Warning über das Schreiben eines Int-Pointers in einen Int. Statement 4 löst eine Warning über das Schreiben eines Char-Pointers in einen Int. Statement 4 ist problematisch, da auf Werte außerhalb der Arraygrenzen zugegriffen wird (das Array ist 10 Ints groß). Dieser Zugriff wird im Gegensatz zu Java auch ausgeführt und überschreibt den an der Stelle befindlichen Wert.

d)* Was macht der folgende Code? Wofür wird `malloc` in C benutzt?

```
int *array123 = malloc(5 * sizeof(int));
```

Warum sollten nicht alle Werte in Variablen abgelegt werden, die in der Funktion direkt deklariert wurden?

`malloc()` reserviert Speicher für fünf zusammenhängende `int`-Werte (auf dem Heap). Die Funktion gibt einen Pointer auf den reservierten Speicherbereich zurück. Pointer auf Speicherbereiche, die im Stackframe der aktuellen Funktion liegen, sollten nicht an die aufrufende Funktion zurückgegeben werden. Der Stackframe der aktuellen Funktion wird freigegeben sobald die Funktion returned, der zurückgegebene Pointer zeigt somit auf einen Speicherbereich der durch spätere Funktionsaufrufe wiederverwendet wird. Daher muss in solchen Fällen mittels `malloc` ein Speicherbereich außerhalb des Stackframes der Funktion allokiert werden, der Pointer auf diesen wird dann zurückgegeben.

e) Welche Rolle spielt der Operator `sizeof()`? Wieso wird an `malloc()` nicht 5 als Parameter übergeben?

`malloc()` erwartet die zu reservierende Speichergröße in Bytes. Unterschiedliche Variablentypen haben unterschiedliche Größen. So hat ein `int` auf x86 meist vier Byte, ein `char` jedoch immer nur ein Byte. `sizeof()` ermittelt die Größe einer Variable oder eines Variablentyps in Byte.

Noch Fragen? Dann schaut doch mal im Zulip-Stream vorbei:
`https://zulip.in.tum.de`

f) Was hat es mit `free()` auf sich? Inwiefern verhält sich Java hier anders als C, wieso musste man diese Funktion in Java nicht nutzen?

Auch Java reserviert Speicher für seine Objekte, verwaltet diesen jedoch automatisch. Der sog. Garbage Collector überprüft regelmäßig, ob ein Objekt noch genutzt wird und gibt dessen Speicher bei Bedarf frei. C besitzt keine Garbage Collection. Gibt der Programmierer ungenutzten reservierten Speicher nicht frei, entsteht ein sog. memory leak. Länger laufende Programme würden so solange Speicher „fressen“, bis kein Arbeitsspeicher mehr frei ist. Dies führt zu einem state namens „out of memory“ und zwangsläufig zum Crash.

Aufgabe 5 *NIX zu finden (Optional/Hausaufgabe)

In diesem Buchstabenfeld sind Befehle aus dem Bereich Unix und Linux versteckt. Wir konnten mehr als 20 Befehle finden, darunter sind zugegebenermaßen allerdings auch einige exotische.
Gelesen werden kann von links nach rechts →, von rechts nach links ←, von oben nach unten ↓ und von unten nach oben ↑. Diagonal haben wir nichts bewusst versteckt, aber vielleicht werden Sie ja trotzdem fündig?

S	L	W	W	D	E	S	E
H	F	C	M	V	H	Y	C
B	O	C	A	N	L	P	N
M	L	R	I	I	A	T	Q
J	D	E	M	R	Y	S	B
O	T	C	A	D	W	P	C
T	R	H	P	K	K	I	N
V	Z	O	D	P	A	X	M

- `at` - führt ein Kommando zu einem späteren Zeitpunkt aus
- `bc` - basic calculator, unterstützt beliebige Präzision
- `cc` - der C-Compiler (meist `gcc` oder `clang`)
- `cp` - kopiert Dateien
- `echo` - gibt Argumente auf stdout aus (meist also auf die Kommandozeile)
- `ed` - bereits 1969 entwickelt, ist dieser Texteditor auch heute noch Teil einer Unix/Linux Installation. Die Bedienung ist gewöhnungsbedürftig.
- `fc` - wird verwendet, um Befehle aus dem Verlauf anzuzeigen und erneut auszuführen
- `fold` - fügt Zeilenumbrüche in Dateien ein, um eine maximale Zeilenbreite zu gewährleisten
- `ld` - ruft den Linker auf, um kompilierte Objektdateien zu einem ausführbaren Programm zu verknüpfen
- `ln` - fügt Referenzen in das Dateisystem ein, um bspw. Verknüpfungen zu erstellen
- `lp` - ursprünglich die Abkürzung für lineprinter ist lp noch heute das Kommando, um Dateien auszudrucken
- `ls` - listet den Inhalt von Verzeichnissen auf
- `mv` - verschiebt Dateien (auch genutzt zum Umbenennen von Dateien)
- `nc` - netcat ist ein Standardwerkzeug, um Netzwerkverbindungen aufzubauen
- `pax` - kann Archive lesen und schreiben und Verzeichnishierarchien kopieren
- `ps` - process status gibt Informationen über laufende Prozesse aus
- `pwd` - print working directory - gibt den Namen des aktuellen Verzeichnisses aus
- `rm` - löscht Dateien
- `sed` - stream editor - muss eine Datei zum Bearbeiten nicht komplett in den Speicher laden
- `sh` - öffnet eine shell (ob dies eine bash ist, hängt vom System ab)
- `tr` - translate - ersetzt bestimmte Zeichen in einer Datei durch andere Zeichen
- `wc` - wordcount zählt Wörter, Zeichen, Zeilen und Bytes in einer Datei und gibt dies aus
- `cd` - wechselt das Verzeichnis (change directory) (diagonal zu finden)