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));
}
```

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.

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.

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

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

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

All Rights Reserved, Do Not Distribute!

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
0b1 1100 0111
0b1100 0000 1101 1110

b)* Hex → Binär

0xAFFE
0xBADE
0xC0FFEE

c)* Dezimal → Hex

123
65
262

d)* Dezimal → Binär

255
99
54

e)* Hex → Dezimal

0xABC
0x64
0x420

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

All Rights Reserved, Do Not Distribute!

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];
```

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)));
```

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);
```

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?

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

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?

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

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

All Rights Reserved, Do Not Distribute!