Programmieren C: Bitoperationen:
Umwandlung “Big Endian” <==> “Little Endian”

Klaus Kusche

Manche Rechner (z.B. IBM Großrechner oder Apple Mac’s, bevor sie auf x86-Prozessoren umgestellt wurden) legen aus mehreren Bytes bestehende Daten (z.B. einen **`int`**) so im Speicher ab, wie wir sie normalerweise anschreiben:
Das höchstwertige Byte (Bit 24 - 31) zuerst, das Byte mit den Bits 0 – 7 als letztes.
Auch am Internet werden alle binären Daten in dieser Reihenfolge übertragen,
man nennt das “*Big Endian*”.

Andere Rechner (z.B. alle Rechner mit Intel- oder ARM-Prozessoren) speichern alle Daten “*Little Endian*”, d.h. das hinterste Byte steht zuerst im Speicher. Die Zahl steht im Speicher also “auf dem Kopf”: Byte 1 ist mit Byte 4 vertauscht und Byte 2 mit Byte 3.

- Verwandle eine einzelne auf der Commandline angegebene Zahl in einen **`int`**.
- Berechne daraus einen zweiten **`int`**,
  in dem die *Bytes wie oben angegeben vertauscht* sind.
- Gib beide *als 8-stellige Hexzahl (mit führenden Nullen)* aus.
  Zur Ausgabe darfst du das *Hex-Format* **`%08X`** von **`printf`** verwenden
  (achtstellig, mit führenden Nullen und großgeschriebenen Hex-Buchstaben).

Zur Big-Endian / Little-Endian-Umwandlung sollst du *nur Bit-Operationen auf* **`int`***’s*
verwenden, keine Multiplikation, Division oder Restbildung, keinen Zugriff
auf einzelne Bytes im Speicher, und auch nicht die dafür auf den meisten Systemen vorhandene vordefinierte Funktion.
Du wirst vermutlich *Bit-Schiebe-Operationen* brauchen. Wie deklarierst du deine **`int`**’s, damit es dabei keine unerwünschten “Überraschungen” gibt?

Tipp: Mit *Hex-Konstanten* wird der Code leichter lesbar und klarer verständlich…

Scharf nachdenken:

- Wie siehst du den beiden Hexzahlen im Output an, ob du richtig gerechnet hast?

Für die Tüftler:

- Die in der Vorlesung gezeigte Variante funktioniert nur für **`int`**’s mit 4 Bytes Länge. Für **`int`**-Typen, die 2 oder 8 Bytes lang sind, müsste man den Code ändern.

  Versuche, statt einer großen Formel für alle 4 Bytes eine *Schleife* zu verwenden, die *pro Umlauf ein Byte* vom ursprünglichen **`int`** in das Ergebnis bringt und für *alle* **`int`**-Längen funktioniert.

  Du kannst mit **`short`** oder **`long long`** testen. Für **`long long`** brauchst du **`atoll(…)`** zum Umwandeln der Eingabe und **`%016llX`** statt **`%08X`** für die Ausgabe. Bei **`short`** funktioniert das normale **`atoi`** und **`%04hX`** in der Ausgabe.

  Die *Größe (in Bytes)* einer Variable **`x`** im Speicher bekommst du mit **`sizeof(x)`**, das solltest du statt einer fixen Anzahl von Bytes in deiner Schleifenbedingung verwenden.