

Speicher

Ein Spion gelangt in den Besitz einer Diskette, die vermutlich geheime Informationen enthält. Was benötigt der Spion, um an die Informationen heranzukommen, die ihn interessieren?

Antwort:

Speicher

Was könnten die Bits

0111 0010 0110 0101 0110 0111 0110 0001 0110 1100 0000 1010 0000 0000 0000 0000
bedeuten?

Speicher

Vergleichen Sie folgende Programmteile. Unterscheiden sich diese semantisch? Wenn ja, wo?

<pre>int sum; int mean(int array[], int count) { int i; for(i=0; i<=count-1; i++) { sum = sum + array[i]; } sum = sum / count; return sum; }</pre>	<pre>float mittel(int zahlen[], int anz) { float summe; for(int nr=0; nr<anz; nr++) { summe += zahlen[nr]; } summe = summe / anz; return summe; }</pre>
--	--

Speicher

Ein PC sendet die (unsigned int) Zahl 12345 an einen Apple Computer.

Was wird auf dem Apple Rechner empfangen, wenn keine besonderen Vorkehrungen getroffen werden? Wie ist das Problem zu lösen?

Hinweis: PCs verwenden little endian, Apple verwendet big endian.

Speicher

Disassemblieren Sie ein Unterprogramm Ihrer Wahl

```
objdump -d test.o | less
```

Vergleichen Sie die Ausgabe mit der entsprechenden von gcc erzeugten Assemblerdatei test.s.

Woher kommen die Unterschiede?

Speicher

Schreiben Sie das folgende Programmfragment so um, dass es statt eines dreidimensionalen Arrays ein eindimensionales Array verwendet!

```
int f[A][B][C];
int a, b, c;

for (a = 0; a < A; a++)
  for (b = 0; b < B; b++)
    for (c = 0; c < C; c++)
      f[a][b][c] = func(a, b, c);
```

Antwort:

Speicher

Wieviel Speicherplatz benötigt die folgende C/C++/Java-Variable list?

```
struct x {
    struct y {
        char name[11];
        short value;
    } entry[100000];
    long nentries;
} list;
```

Antwort:

Speicher

An welchen Adressen liegen die folgenden Variablen, wenn der freie Speicher bei 1238₍₁₆₎ beginnt?

```
char c;
long int x;
struct {
    char tab;
    long value;
} member[2];
```

Antwort:

Speicher

Wieviel Speicherplatz benötigt die folgende C/C++/Java-Variable `list` mit korrektem Alignment?

```
struct {
    struct {
        char name[11];
        short value;
    } entry[100000];
    long nentries;
} list;
```

Antwort:

Speicher

Wie kann ein Compiler/Assembler/Linker folgende Variablen mit korrektem Alignment und möglichst wenig Speicher-Overhead speichern?

```
char c;
long int x;
short int y;
char c2;
long int z;
```

Antwort:

Speicher

Im Speicher sollen Datensätze der Form

```
struct datensatz {  
    char name[12];  
    long int punkte;  
};
```

gespeichert werden (Größe char: 1 Byte, long int: 4 Byte). (Ehemalige Klausuraufgabe.)

a) Wieviele solcher Strukturen lassen sich in 16 KByte speichern?

b) Warum bringt es auf einem 32-Bit-Rechner unter Umständen nichts, die Anzahl der Zeichen für den Namen (gespeichert in „name“) von 12 auf 10 zu erniedrigen, wenn Speicherplatz gespart werden soll?

Speicher

Antwort zu a):

Antwort zu b):

Speicher

Gegeben sei die folgende Datenstruktur eines 32-Bit-Rechners

```
struct student {  
    char name[10];  
    int punkte;  
}
```

und der folgende Speicherauszug (im 16er-System)

```
1ab0: 00 00 34 00 00 00 00 32   4D 55 45 4C 4C 45 52 00  
1ac0: 00 00 FE 12 00 00 00 2E   4D 45 49 45 52 00 00 00
```

Wie heisst der Student, dessen Datensatz an der Adresse 1ab8 mit korrektem Alignment gespeichert ist? Wieviele Punkte hat er?

Hinweis: Name ist im ASCII Format gespeichert ('A'=41, 'B'=42, ... 'E'=45, 'L'=4C, 'R'=52)