3.3 一级指针简介_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 3.3 一级指针简介涵盖海量编程基础技术教程，以图文图表的形式，把晦涩难懂的编程专业用语，以通俗易懂的方式呈现给用户。

3. 一级指针简介

int m = 20;

int *p = &m;

指针变量 p 本身的数据类型：int *

指针变量保存的对象的数据类型： int

```
不同系统使用的CPU不同，对数据的存储形式也有不同，成分为以下两种。

大端模式：ARM，摩托罗拉

小端模式：intel,MIPS

=====================================================

大端模式：内存的高地址存储数据的低位，内存的低地址存储数据的高位。
(低地址存高位---低对高)

小端模式：内存的低地址存储数据的低位，内存的高地址存储数据的高位。
(低地址存低位---低对低)

          int x = 0x12345678;


              小端模式                大端模式
低地址
0xdff30       0x78                    0x12              <------
0xdff31       0x56                    0x34
0xdff32       0x34                    0x56
0xdff33       0x12                    0x78
高地址
```

- ## 指针的结论
  - 在 32bit 的操作系统中，所有类型的指针变量都是 4bytes. [因为地址为 4bytes]

示例代码：

```
#include <stdio.h>

int main()
{
        char *x;
        short *y;
        int *z;

        printf("sizeof(x) = %d\n",sizeof(x));
        printf("sizeof(y) = %d\n",sizeof(y));
        printf("sizeof(z) = %d\n",sizeof(z));


        return 0;
}
```

编译方法：

运行结果:

```
sizeof(x) = 4
sizeof(y) = 4
sizeof(z) = 4
```

- 不同类型的指针变量，对 C 语言中的同一块内存进行读取的时候，每次读取的字节数不同。(读取为指针变量 + *, 剩下数据类型的大小)，具体如下:

```
int a = 0x12345678;  (ubuntu默认小端模式)

低地址
0xdff00                0x78
0xdff01                0x56
0xdff02     0x34
0xdff03                0x12
高地址

 char *p = (char *)&a;
 short *q = (short *)&a;
 int *m = &a;

 *p;
 *q;
 *m;
```

示例代码:

```
#include <stdio.h>

int main()
{
        char *x;
        short *y;
        int *z;
        int t = 0x12345678;

        x = (char *)&t;
        y = (short *)&t;
        z = &t;

        printf("*x = %#x\n",*x);
        printf("*y = %#x\n",*y);
        printf("*z = %#x\n",*z);
```

运行结果:

```
*x = 0x78
*y = 0x5678
*z = 0x12345678
```

- 在 32bit 的操作系统中，不同类型的指针变量每次的移动大小不一样。(每次移动的大小为指针变量 + *, 剩下数据类型的大小)，具体如下:

```
 int a = 0x12345678;
 char *p = (char *)a;
 short *q = (short *)a;
 int *m = a;

 p++;
 q++;
 m++;
```

示例代码:

```
#include <stdio.h>

int main()
{
        char *x;
        short *y;
        int *z;
```

```
        int t = 0x12345678;

        x = (char *)&t;
        y = (short *)&t;
        z = &t;




        printf("&t = %p\n",&t);
        printf("x = %p\n",x);
        printf("y = %p\n",y);
        printf("z = %p\n",z);

        printf("======================\n");

        x++;
        y++;
        z++;

        printf("&t = %p\n",&t);
        printf("x = %p\n",x);
        printf("y = %p\n",y);
        printf("z = %p\n",z);

        return 0;
}
```

运行结果：

```
&t = 0xff8734dc
x = 0xff8734dc
y = 0xff8734dc
z = 0xff8734dc
======================
&t = 0xff8734dc
x = 0xff8734dd
y = 0xff8734de
z = 0xff8734e0
```

定义一个数组
int a[5] = {0};
要求大家从键盘上输入数据给数组赋值。
然后定义一个指针int *p_max要求它保存最大值的地址。
然后通过*p_max输出最大值。

```
unsigned int data = 0x11223344;
unsigned short *q = NULL;

unsigned short t1 = 0;
unsigned short t2 = 0;

(1)要求指针q保存data的地址.

(2)要求利用q读取data的低2个字节赋值給t1 ===>0x3344
   要求利用q读取data的高2个字节赋值給t2 ===>0x1122

(3)输出t1和 t2 的和与差
```

全文完