

Windows和Ubuntu下内存地址分配问题

2020年12月26日 17:29

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

```
int pointer_test()
```

```
{
```

```
    int ia;
    int *pi;
    char *pc;
```

```
    printf("&ia = %p\n", &ia);
    printf("&pi = %p\n", &pi);
    printf("&pc = %p\n", &pc);
```

```
    ia = 0x12345678;
    pi = &ia;
    pc = (char *)&ia;
    printf("ia = 0x%x\n", ia);
    printf("pi = %p\n", pi);
    printf("pc = %p\n", pc);
    printf("len(int) = %d\n", sizeof(int));
    printf("len(int *) = %d\n", sizeof(int *));
    printf("len(char *) = %d\n", sizeof(char *));
```

```
    printf("*pi = 0x%x\n", *pi);
    printf("pc = %p\t", pc);    printf("*pc = 0x%x\n", *pc); pc = pc + 1;
    printf("pc = %p\t", pc);    printf("*pc = 0x%x\n", *pc); pc = pc + 1;
    printf("pc = %p\t", pc);    printf("*pc = 0x%x\n", *pc); pc = pc + 1;
    printf("pc = %p\t", pc);    printf("*pc = 0x%x\n", *pc);
    return 0;
```

```
}
```

```
int main()
```

```
{
```

```
    pointer_test();
```

```

system("pause");
return 0;
}

```

使用上述代码时，会发现Windows下使用visual studio17和Ubuntu的输出不同。具体如下。

运行结果

visual studio17(debug模式 x86)



```

E:\VS2017CODE\Ctest\Debug\Ctest.exe
&ia = 010FF9A8
&pi = 010FF99C
&pc = 010FF990
ia = 0x12345678
pi = 010FF9A8
pc = 010FF9A8
len(int) = 4
len(int *) = 4
len(char *) = 4
*pi = 0x12345678
pc = 010FF9A8      *pc = 0x78
pc = 010FF9A9      *pc = 0x56
pc = 010FF9AA      *pc = 0x34
pc = 010FF9AB      *pc = 0x12
请按任意键继续. . .

```

visual studio17(release模式 x86)



```

E:\VS2017CODE\Ctest\Release\Ctest.exe
&ia = 0078FD5C
&pi = 0078FD54
&pc = 0078FD58
ia = 0x12345678
pi = 0078FD5C
pc = 0078FD5C
len(int) = 4
len(int *) = 4
len(char *) = 4
*pi = 0x12345678
pc = 0078FD5C      *pc = 0x78
pc = 0078FD5D      *pc = 0x56
pc = 0078FD5E      *pc = 0x34
pc = 0078FD5F      *pc = 0x12
请按任意键继续. . .

```

ubuntu12(32位 关闭地址随机化)

```
[12/26/2020 01:13] seed@ubuntu:~/CSAPPS$ gcc -o pointer_int_char pointer_i
.c
[12/26/2020 01:13] seed@ubuntu:~/CSAPPS$ ./pointer_int_char
&ia = 0xbffff374
&pi = 0xbffff378
&pc = 0xbffff37c
ia = 0x12345678
pi = 0xbffff374
pc = 0xbffff374
len(int) = 4
len(int *) = 4
len(char *) = 4
*pi = 0x12345678
pc = 0xbffff374 *pc = 0x78
pc = 0xbffff375 *pc = 0x56
pc = 0xbffff376 *pc = 0x34
pc = 0xbffff377 *pc = 0x12
[12/26/2020 01:13] seed@ubuntu:~/CSAPPS$
```

ubuntu18(64位 未关闭地址随机化)

```
misstory@misstory:~/CSAPPS$ ./pointer_int_char
&ia = 0x7fffa7854614
&pi = 0x7fffa7854618
&pc = 0x7fffa7854620
len(int) = 4
len(int *) = 8
len(char *) = 8
ia = 0x12345678
pi = 0x7fffa7854614
pc = 0x7fffa7854614
*pi = 0x12345678
pc = 0x7fffa7854614 *pc = 0x78
pc = 0x7fffa7854615 *pc = 0x56
pc = 0x7fffa7854616 *pc = 0x34
pc = 0x7fffa7854617 *pc = 0x12
```

思考

- ? windows中均使用x86，即32位编译器。release模式下，变量的地址是先减少8字节后增加4字节，看起来很奇怪，内存空间地址分布由高到低分别是ia->pc->pi，不符合变量定义的顺序 (ia->pi->pc)；删减上述代码，使其仅保留ia, pc, pi的定义语句及输出语句，结果如下。可以看出地址是递减4字节，符合预期。猜测可能是编译器的策略问题，但是具体原因不明白。

```
E:\VS2017CODE\Ctest\Release\Ctest.exe
&ia = 00CFFDC4
&pi = 00CFFDC0
&pc = 00CFFDBC
请按任意键继续. . .
```

debug模式下，地址递减12字节，为什么不是4字节。网上查阅后，得知，debug模式下，编译器会插入一些其他信息。

Ubuntu中，32位系统是正常递增4字节，符合逻辑；64位系统中，第一次增加4字节，第二次增加8字节，符合逻辑。

这样看起来，似乎windows和ubuntu在对局部变量定义时，地址空间的分配策略不同。Windows把先定义的变量放在高地址，后定义的变量放在低地址，即是由高地址向低地址分配。Ubuntu把先定义的变量放在低地址，后定义的变量放在高地址，即是由低地址向高地址分配。相同的是，每个变量自己的地址增长顺序都是从低到高，如下分别时Windows下和Linux下运行的结果。关于分配规律的介绍可见[浅谈windows和linux下内存分配规律_点滴-CSDN博客](#)。

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

int pointer_test()
{
    int ia;
    int *pi;
    char *pc;
    int a[3];

    printf("&ia = %p\n", &ia);
    printf("&pi = %p\n", &pi);
    printf("&pc = %p\n", &pc);
    printf("&a[0] = %p\n", &a[0]);
    printf("&a[1] = %p\n", &a[1]);
    printf("&a[2] = %p\n", &a[2]);
}
```

```
E:\VS2017CODE\Ctest\Release\Ctest.exe
&ia = 0055FBB4
&pi = 0055FBB0
&pc = 0055FBAC
&a[0] = 0055FBB8
&a[1] = 0055FBBc
&a[2] = 0055FBC0
请按任意键继续. . .
```

```
.C
[12/26/2020 23:23] seed@ubuntu:~/CSAPPS$ ./pointer_int_char
&ia = 0xbffff374
&pi = 0xbffff378
&pc = 0xbffff37c
&a[0] = 0xbffff368
&a[1] = 0xbffff36c
&a[2] = 0xbffff370
[12/26/2020 23:23] seed@ubuntu:~/CSAPPS$
```

关于指针长度，32位系统的指针长度都是4字节，64位系统的指针长度都是8字节。