# Union

by 刘驭壬

01
什么是Union

02
对Union的认识

03
Union的使用方法
和用途

# Content

01

# PART ONE 什么是Union

# Union的定义

Weki: **union** is a value that may have any of several representations or formats; or it is a data structure that consists of a variable that may hold such a value.
Some programming languages support special data types, called **union types**, to describe such values and variables. In other words, a union type definition will specify which of a number of permitted primitive types may be stored in its instances

百度百科：“联合”是一种特殊的类，也是一种构造类型的数据结构。在一个“联合”内可以定义多种不同的数据类型，一个被说明为该“联合”类型的变量中，允许装入该“联合”所定义的任何一种数据，这些数据共享同一段内存，以达到节省空间的目的（还有一个节省空间的类型：位域）。 这是一个非常特殊的地方，也是联合的特征。另外，同 struct 一样，联合默认访问权限也是公有的，并且，也具有成员函数。

定义简单的Union类型数据

```
Union  student
{
        char name[12];
        unsigned stunum;
        int age;
};
```

```
Struct student
{
        char name[12];
        unsigned stunum;
        int age;
};
```

Few things are impossible in themselves; and it is often for want of will, rather than of means, that man fails to succeed.

02

# PART TWO 对Union的认识

union 维护足够的空间来置放多个数据成员中的“一种”，而不是为每一个数据成员配置空间，在union 中所有的数据成员共用一个空间，同一时间只能储存其中一个数据成员，所有的数据成员具有相同的起始地址。

一个union 只配置一个足够大的空间以来容纳最大长度的数据成员。

union在操作系统底层的代码中用的比较多，因为它在内存共赏布局上方便且直观。

# 关于Union的几个例子

```
#include<iostream>
using namespace std;

typedef struct
{
    char name[12];
    unsigned stunum;
    int age;
}Student1;

typedef union
{
    char name[12];
    unsigned stunum;
    int age;
}Student2;

int main()
{
    cout<<"The size of struct Student is "<<sizeof(Student1)<<endl;
    cout<<"The size of union  Student is "<<sizeof(Student2)<<endl;
    return 0;
}
```

```
The size of struct Student is 20
The size of union  Student is 12

--------------------------------

Process exited after 0.2669 seconds with return value 0
请按任意键继续. . .
```

03

PART THREE

# Union 的使用方法和用途

```cpp
#include<iostream>
#include<stdio.h>
using namespace std;

union convert
{
    int a;
    unsigned b;
};

int main()
{
    convert test;

    test.a=1;
    cout<<"test.a="<<test.a<<endl;
    cout<<"test.b="<<test.b<<endl;

    test.a=-1;
    cout<<"test.a="<<test.a<<endl;
    cout<<"test.b="<<test.b<<endl;
    printf("test.a的十六进制编码为%x\ntest.b的十六进制编码为%x\n",test.a,test.a);

    return 0;
}
```

```
test.a=1
test.b=1
test.a=-1
test.b=4294967295
test.a的十六进制编码为ffffffff
test.b的十六进制编码为ffffffff

--------------------------------
```

# 匿名Union

匿名union是没有名称和声明列表的union,它的声明形式如下:
union { member-list } ;

匿名union仅仅通知编译器它的成员变量共享一个地址,而变量本身是直接引用的,不使用通常的点号运算符语法.也正因此,匿名union与同一程序块内的其它变量具有相同的作用域级别,需注意命名冲突.

以下内容如果听不懂可以不管，暂时用不到！

# 匿名Union

```
17 typedef struct {
18     union
19     {
20         union {
21             uint32_t _32;
22             uint16_t _16;
23             uint8_t _8[2];
24         } gpr[8];
25
26         /* Do NOT change the order of the GPRs' definitions. */
27         struct
28         {
29             uint32_t eax, ecx, edx, ebx, esp, ebp, esi, edi;
30         };
31     };
32     swaddr_t eip;
33     struct
34     {
35         uint32_t CF:1;
36         uint32_t rsvd1:1;
37         uint32_t PF:1;
38         uint32_t rsvd2:1;
39         uint32_t AF:1;
40         uint32_t rsvd3:1;
41         uint32_t ZF:1;
42         uint32_t SF:1;
43         uint32_t TF:1;
44         uint32_t IF:1;
45         uint32_t DF:1;
46         uint32_t OF:1;
47         uint32_t OL:1;
48         uint32_t IP:1;
49         uint32_t NP:1;
50         uint32_t rsvd4:1;
51         uint32_t RF:1;
52         uint32_t VM:1;
53         uint32_t rsvd5:14;
54     }eflags;
55
56 } CPU_state;
```

Thanks for your listening