

Union Type

TPV2
Samir Genaim

¿Qué es Union Type?

```
void foo(bool f) {  
    int x;  
    float z;  
  
    if (f) {  
        x = 1;  
        // do something with x  
    } else {  
        z = 2.4f;  
        // do something with z  
    }  
}
```

A veces, hay variables que no se usan a la vez durante la ejecución, se usa uno o otro depende de alguna condición

Usa sólo x

Usa sólo z

- ♦ `x` y `z` ocupan `sizeof(int)+sizeof(float)` bytes en la memoria
- ♦ Si podemos decir al compilador que vamos a usar sólo una a la vez, lo que puede hacer es usar la misma dirección en la memoria para ambos y así ocupar sólo `MAX(sizeof(int),sizeof(float))` bytes
- ♦ Union Types nos permite transmitir esa información al compilador

Ejemplo

```
void foo(bool f) {  
    union {  
        int x;  
        float z;  
    }
```


← x y z ocupan la misma misma memoria.

```
    if (f) {  
        x = 1;  
        // do something with x  
    } else {  
        z = 2.4f;  
        // do something with z  
    }  
}
```

La responsabilidad es tuya ...

```
void foo(bool f) {  
    union {  
        int x;  
        float z;  
    }
```

La responsabilidad de uso correcto es tuya, si asignas uno y usas el otro puedes tener resultados inesperados



```
    x = 3;  
    cout << "x = " << x << endl;  
    cout << "z = " << z << endl;  
}
```

x = 3

z = 4.2039e-45

Union y Struct

```
struct SomeType_1 {  
    int id;  
    float x;  
    int z;  
};
```

```
struct SomeType_2 {  
    int id;  
    union {  
        float x;  
        int z;  
    };  
};
```

```
void foo() {  
    cout << "sizeof(SomeType_1) = " << sizeof(SomeType_1) << endl;  
    cout << "sizeof(SomeType_2) = " << sizeof(SomeType_2) << endl;  
}
```



depende del valor de id, se usa el atributo x o z

```
sizeof(SomeType_1) = 12  
sizeof(SomeType_2) = 8
```

Union y Struct

Lo vimos con el patron event queue ...

```
struct SoundEventType { PLAYMUSIC, PLAYSOUND, ... };
```

```
struct SoundEvent {  
    SoundEventType type_;
```

```
    union {
```

```
        Resources::MusicId musicId_;
```

```
        Resources::SoundEffectId soundId_;
```

```
    };
```

```
};
```

Musica o Sonido



depende del valor de type_ se was musicId_ o soundId_



Union para Mensajes en el ECS

Se puede usar para definir el tipo de todos **Message** que usamos en ECS. Un mensaje siempre tiene **id_** y depende de ese valor usado **v1**, o **v2**, etc.

```
struct Message {  
    Uint8 id_;  
    union {  
        MsgType1 v1;  
        MsgType2 v2;  
        ...  
    };  
};
```

```
struct MsgType1 {  
    Uint8 side_;  
};
```

```
struct MsgType2 {  
    Uint8 winner_;  
};
```

...

Cuidado al incluir algo que no es de "primitive type", en ese caso hay que definir constructoras por defecto y de copia casi en todas partes ...