

Structures and Unions

Young W. Lim

2017-02-09 Thr

1 Introduction

- References
- Structure Background
- Union Background

"Self-service Linux: Mastering the Art of Problem Determination", Mark Wilding
"Computer Architecture: A Programmer's Perspective", Bryant & O'Hallaron

I, the copyright holder of this work, hereby publish it under the following licenses: GNU head Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled GNU Free Documentation License.

CC BY SA This file is licensed under the Creative Commons Attribution ShareAlike 3.0 Unported License. In short: you are free to share and make derivative works of the file under the conditions that you appropriately attribute it, and that you distribute it only under a license compatible with this one.

- creating data types
- by combining objects of different types
- aggregate multiple objects into a single unit
- group objects possible different types into a single object
- like arrays
 - stored in a contiguous region
 - a pointer to a structure : the address of its 1st byte
- each field accessed by offsets
- displacements in memory referencing instruction

Structure Declaration (1)

```
struct rec {  
    int i;  
    int j;  
    int a[3];  
    int *p;  
};  
  
0x00 : i  
0x04 : j  
0x08 : a[0]  
0x0C : a[1]  
0x10 : a[2]  
0x14 : p  
0x1C :
```

- allow an object to be referenced in multiple ways
- using several different types
- rather than having the different references point to different blocks
- but they all reference the same block
- the use of two different fields is mutually exclusive
- can reduce memory usage
- can be used to access the bit patterns of different data types

Union Declaration (1)

```
struct S3 {  
    char c;  
    int i[2];  
    double v;  
};
```

```
0x00 : c  
0x04 : i[0]  
0x08 : i[1]  
0x0c : v  
0x20 :
```

```
union U3 {  
    char c;  
    int i[2];  
    double v;  
}
```

```
0x00 : c  
0x00 : i[0]  
0x00 : i[1]  
0x00 : v  
0x08 :
```

Union Declaration (2)

```
struct NODE {
    struct NODE *left;
    struct NODE *right;
    double data;
};

union NODE{
    struct NODE {
        struct NODE *left;
        struct NODE *right;
    } internal;
    double data;
};
```

```
struct NODE {
    int is_leaf;
    union NODE{
        struct NODE {
            struct NODE *left;
            struct NODE *right;
        } internal;
        double data;
    } info;
};
```