Structures and Unions

Young W. Lim

2017-02-09 Thr

Outline

Based on

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

Strudctures

- 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 :
```

Unions

- allow an object to be referenced in mulitple ways
- using several different types
- rather than having the different reference different blocks
- but they all reference the same block
- the use of two different fields is mutually exclusive
- can reduce memory usage3
- 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;
};
```