BIT-FIELDS & UNIONS

Dr. H. IREM TURKMEN

Bit – fields of Structures

We can specify size of structure members in bits in order to use memory efficiently

- bitField_type bitField_name **:** numberOfBits

```
typedef struct
{
unsigned int day:5; //0 – 25-1 (0-31)
unsigned int month:4;// 0 – 24-1 (0-15)
unsigned int year:11; // 0 – 211-1 (0-2047)
}DATE;
// Approximately 20 bits
(may be a bit further since gaps may occur)
```

```
typedef struct
{
unsigned int day;
unsigned int month;
unsigned int year;
}DATE;
//3*32 bits for a 32 bit system
```

Bit – fields of Structures

The allowable data types for a bit field include unsigned int and signed int

• YOU CAN NOT!!

▫ define an array of bit fields

(but you can define arrays of structures that have bitfields)

▫ use bit fields greater than size of int

(int too_long:40 //NOT OK for a 32 bit system)

▫ take the address of a bit field (using & is not allowed. Use a temporary variable in order to use scanf())

▫ have a pointer to a bit field

Bit – fields of Structures

How many bits do you need to code height of a person (in cm)?

- How many bits do you need to code gender of a person?

```
typedef struct
{
        unsigned int height:8; //0 – 28-1 (0-255)
        //or unsigned char height;
        unsigned int gender:1; // 0 – 1 (male/female)
}PERSON;
```

Unions

Unions are similar to structures except that the members are overlaid one on top of another, so members share the same memory.

! There are two basic applications for unions:

◦ Interpreting the same memory in different ways.

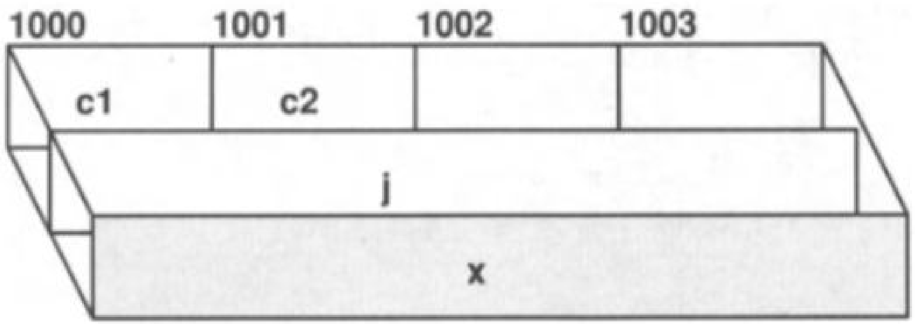
◦ Creating flexible structures that can hold different types of data.

Unions

- Example:

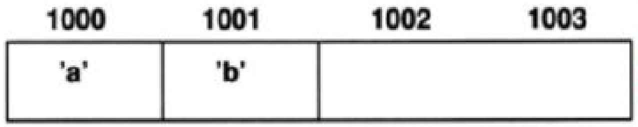
```
typedef union
{
  struct
  {
    char c1, c2;
  } s;
  long j;
  float x;
} U;

U example;
```



- Usage:

```
example.s.c1 = 'a';
example.s.c2 = 'b';
```



* If you make the assignment:
example.j = 5; // it overwrites the 2 chars, using all 4 bytes to store value 5.

Initializing Unions

```
union init_example
{
  int i;
  float f;
}

union init_example test={1};
```

```
union u
{
  struct {int i; float f;} s;
  char ch[6];
}

union u test2={1,1.0};
union u test2={.ch={1,2,3,4,5,6}};
```