16.4 Unions

A *union*, like a structure, consists of one or more members, possibly of different types. However, the compiler allocates only enough space for the largest of the members, which overlay each other within this space. As a result, assigning a new value to one member alters the values of the other members as well.

To illustrate the basic properties of unions, let's declare a union variable, u, with two members:

```
union {
  int i;
  double d;
} u;
```

Notice how the declaration of a union closely resembles a structure declaration:

```
struct {
  int i;
  double d;
} s;
```

In fact, the structure s and the union u differ in just one way: the members of s are stored at *different* addresses in memory, while the members of u are stored at the *same* address. Here's what s and u will look like in memory (assuming that int values require four bytes and double values take eight bytes):