

## Using Unions to Save Space

We'll often use unions as a way to save space in structures. Suppose that we're designing a structure that will contain information about an item that's sold through a gift catalog. The catalog carries only three kinds of merchandise: books, mugs, and shirts. Each item has a stock number and a price, as well as other information that depends on the type of the item:

*Books:* Title, author, number of pages

*Mugs:* Design

*Shirts:* Design, colors available, sizes available

Our first design attempt might result in the following structure:

```
struct catalog_item {
    int stock_number;
    double price;
    int item_type;
    char title[TITLE_LEN+1];
    char author[AUTHOR_LEN+1];
    int num_pages;
    char design[DESIGN_LEN+1];
    int colors;
    int sizes;
};
```

The `item_type` member would have one of the values `BOOK`, `MUG`, or `SHIRT`. The `colors` and `sizes` members would store encoded combinations of colors and sizes.

Although this structure is perfectly usable, it wastes space, since only part of the information in the structure is common to all items in the catalog. If an item is a book, for example, there's no need to store `design`, `colors`, and `sizes`. By putting a union inside the `catalog_item` structure, we can reduce the space required by the structure. The members of the union will be structures, each containing the data that's needed for a particular kind of catalog item:

```
struct catalog_item {
    int stock_number;
    double price;
    int item_type;
    union {
        struct {
            char title[TITLE_LEN+1];
            char author[AUTHOR_LEN+1];
            int num_pages;
        } book;
        struct {
            char design[DESIGN_LEN+1];
        } mug;
    };
};
```