```
struct {
  char *cmd_name;
  void (*cmd_pointer)(void);
} file_cmd[] =
 {{"new",        new_cmd},
  {"open",       open_cmd},
  {"close",      close_cmd},
  {"close all", close_all_cmd},
  {"save",       save_cmd},
  {"save as",    save_as_cmd},
  {"save all",   save_all_cmd},
  {"print",      print_cmd},
  {"exit",       exit_cmd}
 };
```

Programming Projects

1. Modify the `inventory.c` program of Section 16.3 so that the `inventory` array is allocated dynamically and later reallocated when it fills up. Use `malloc` initially to allocate enough space for an array of 10 `part` structures. When the array has no more room for new parts, use `realloc` to double its size. Repeat the doubling step each time the array becomes full.

2. Modify the `inventory.c` program of Section 16.3 so that the `p` (print) command calls `qsort` to sort the `inventory` array before it prints the parts.

3. Modify the `inventory2.c` program of Section 17.5 by adding an `e` (erase) command that allows the user to remove a part from the database.

4. Modify the `justify` program of Section 15.3 by rewriting the `line.c` file so that it stores the current line in a linked list. Each node in the list will store a single word. The `line` array will be replaced by a variable that points to the node containing the first word. This variable will store a null pointer whenever the line is empty.

5. Write a program that sorts a series of words entered by the user:

```
Enter word: foo
Enter word: bar
Enter word: baz
Enter word: quux
Enter word:

In sorted order: bar baz foo quux
```

   Assume that each word is no more than 20 characters long. Stop reading when the user enters an empty word (i.e., presses Enter without entering a word). Store each word in a dynamically allocated string, using an array of pointers to keep track of the strings, as in the `remind2.c` program (Section 17.2). After all words have been read, sort the array (using any sorting technique) and then use a loop to print the words in sorted order. *Hint:* Use the `read_line` function to read each word, as in `remind2.c`.

6. Modify Programming Project 5 so that it uses `qsort` to sort the array of pointers.

7. (C99) Modify the `remind2.c` program of Section 17.2 so that each element of the `reminders` array is a pointer to a `vstring` structure (see Section 17.9) rather than a pointer to an ordinary string.