

Chapter 9

Strings

Figure 9.1 Right and Left Justification of Strings

Right-Justified	Left-Justified
George Washington	George Washington
John Adams	John Adams
Thomas Jefferson	Thomas Jefferson
James Madison	James Madison

Figure 9.2 String Input/Output with scanf and printf

```

1. #include <stdio.h>
2.
3. #define STRING_LEN 10
4.
5. int
6. main(void)
7. {
8.     char dept[STRING_LEN];
9.     int course_num;
10.    char days[STRING_LEN];
11.    int time;
12.
13.    printf("Enter department code, course number, days and ");
14.    printf("time like this:\n> COSC 2060 MHF 1410\n");
15.    scanf("%s%d%s", dept, &course_num, days, &time);
16.    printf("to %d meets %s at %d\n", dept, course_num, days, time);
17.
18.    return (0);
19. }

```

Enter department code, course number, days and time like this:
> COSC 2060 MHF 1410
> MATH 1270 TR 800
MATH 1270 meets TR at 800

Figure 9.3 Execution of scanf ("%s", dept);

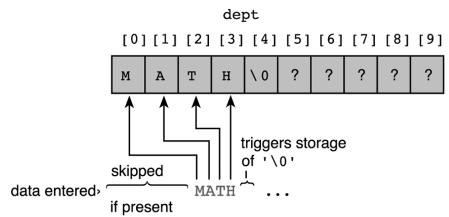


Figure 9.4 Execution of scanf("%s%d%s%d", dept, &course_num, days, &time); on Entry of Invalid Data

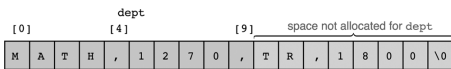


Figure 9.5 Execution of strncpy(result, s1, 9);

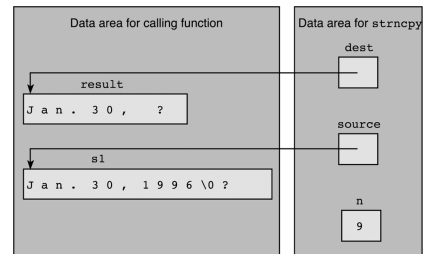


Figure 9.6 Execution of `strncpy(result, &s1[5], 2);`

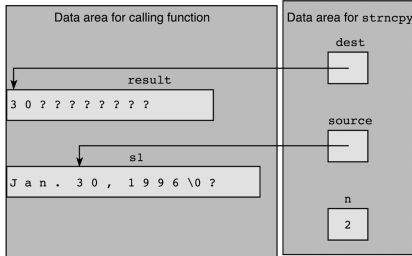


Figure 9.7 Program Using `strncpy` and `strcpy` Functions to Separate Compounds into Elemental Components

```

1  /* Displays each elemental component of a compound
2  */
3
4  #include <stdio.h>
5  #include <string.h>
6  #define ONE_LEN 10 /* size of string to hold a compound */
7  #define ELEM_LEN 10 /* size of string to hold a component */
8
9  int
10 main(void)
11 {
12     char compound[ONE_LEN]; /* string representing a compound */
13     char elem[ELEM_LEN]; /* one elemental component */
14     int i;
15
16     /* Get data string representing compound */
17     printf("Enter a compound: ");
18     scanf("%s", compound);
19
20     /* Displays each elemental component. There are identified
21     by an initial capital letter. */
22     if (strlen(compound) > 0)
23     for (i = 0; i < strlen(compound); i++)
24     if (compound[i] == 'A' || compound[i] == 'a') {
25         strcpy(elem, compound[i]);
26         printf("Element: %s\n", elem);
27         i++;
28     }
29
30     /* Displays the last component
31     printf("Last: %s\n", strcpy(elem, compound[i]));
32
33     return (0);
34 }
35
36 Make a compound: H2O4
37
38
39

```

Figure 9.8 Demonstration of Whole-Line Input

```

1  /*
2  * Numbers and double spaces lines of a document. Lines longer than
3  * LINE_LEN - 1 characters are split on two lines.
4  */
5
6  #include <stdio.h>
7  #include <string.h>
8
9  #define LINE_LEN 80
10 #define NAME_LEN 40
11
12 int
13 main(void)
14 {
15     char line[LINE_LEN], inname[NAME_LEN], outname[NAME_LEN];
16     FILE *inp, *outp;
17     char *status;
18     int i = 0;
19
20     printf("Name of input file: ");
21     scanf("%s", inname);
22     printf("Name of output file: ");
23     scanf("%s", outname);
24
25     inp = fopen(inname, "r");
26     outp = fopen(outname, "w");
27
28     for (status = fgets(line, LINE_LEN, inp);
29         status != 0;
30         status = fgets(line, LINE_LEN, inp)) {
31         if (line[strlen(line) - 1] == '\n')
32             line[strlen(line) - 1] = '\0';
33         printf(outp, "%30s %s\n", i++, line);
34     }
35 }
36
37 (continued)

```

Figure 9.8 Demonstration of Whole-Line Input

```

35. return (0);
36. }

```

File used as input

In the early 1960s, designers and implementers of operating systems were faced with a significant dilemma. As people's expectations of modern operating systems escalated, so did the complexity of the systems themselves. Like other programmers solving difficult problems, the systems programmers desperately needed the readability and modularity of a powerful high-level programming language.

Output file

- 1> In the early 1960s, designers and implementers of operating
- 2> systems were faced with a significant dilemma. As people's
- 3> expectations of modern operating systems escalated, so did
- 4> the complexity of the systems themselves. Like other
- 5> programmers solving difficult problems, the systems
- 6> programmers desperately needed the readability and
- 7> modularity of a powerful high-level programming language.

Figure 9.9 Numeric and String Versions of Portions of Selection Sort That Compare and Exchange Elements

```

Comparison (in function that finds index of "smallest" remaining element)
Numeric      String
if (list[i] < list[first])    if (strcmp(list[i], list[first]) < 0)
    first = i;                first = i;

Exchange of elements
temp = list[index_of_min];    strcpy(temp, list[index_of_min]);
list[index_of_min] = list[fill];  strcpy(list[index_of_min], list[fill]);
list[fill] = temp;           strcpy(list[fill], temp);

```

Figure 9.10 Sentinel-Controlled Loop for String Input

```

1. printf("Enter list of words on as many lines as you like.\n");
2. printf("Separate words by at least one blank.\n");
3. printf("When done, enter is to quit.\n", SENT);
4.
5. for (scanf("%s", word);
6.     strcmp(word, SENT) != 0;
7.     scanf("%s", word)) {
8.     /* process word */
9.     ...
10. }

```

Figure 9.11 Exchanging String Elements of an Array

```

1. strcpy(temp, list[index_of_min]);
2. strcpy(list[index_of_min], list[fill]);
3. strcpy(list[fill], temp);

```

Figure 9.12 Executing strcpy (list [index_of_min], list[fill]);

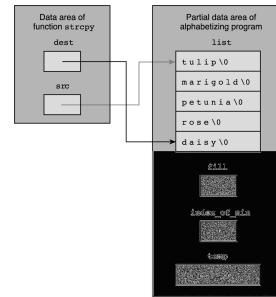


Figure 9.13 An Array of Pointers

