1.3 Protocol Independence

Our program in Figure 1.5 is *protocol-dependent* on IPv4. We allocate and initialize a `sockaddr_in` structure, we set the family of this structure to `AF_INET`, and we specify the first argument to `socket` as `AF_INET`.

To modify the program to work under IPv6, we must change the code. Figure 1.6 shows a version that works under IPv6, with the changes highlighted in bold.

Figure 1.6 Version of Figure 1.5 for IPv6.

intro/daytimetcpcliv6.c

```
 1 #include     "unp.h"

 2 int
 3 main(int argc, char **argv)
 4 {
 5     int     sockfd, n;
 6     char    recvline[MAXLINE + 1];
 7     struct sockaddr_in6 servaddr;

 8     if (argc != 2)
 9         err_quit("usage: a.out <IPaddress>");

10     if ( (sockfd = socket(AF_INET6, SOCK_STREAM, 0)) < 0)
11         err_sys("socket error");

12     bzero(&servaddr, sizeof(servaddr));
13     servaddr.sin6_family = AF_INET6;
14     servaddr.sin6_port = htons(13);    /* daytime server */
15     if (inet_pton(AF_INET6, argv[1], &servaddr.sin6_addr) <= 0)
16         err_quit("inet_pton error for %s", argv[1]);

17     if (connect(sockfd, (SA *) &servaddr, sizeof(servaddr)) < 0)
18         err_sys("connect error");

19     while ( (n = read(sockfd, recvline, MAXLINE)) > 0) {
20         recvline[n] = 0;    /* null terminate */
21         if (fputs(recvline, stdout) == EOF)
22             err_sys("fputs error");
23     }
24     if (n < 0)
25         err_sys("read error");

26     exit(0);
27 }
```

Only five lines are changed, but what we now have is another protocol-dependent program; this time, it is dependent on IPv6. It is better to make a program *protocol-independent*. Figure 11.11 will show a version of this client that is protocol-independent by using the `getaddrinfo` function (which is called by `tcp_connect`).

Another deficiency in our programs is that the user must enter the server's IP address as a dotted-decimal number (e.g., 206.168.112.219 for the IPv4 version). Humans work better with names instead of numbers (e.g., `www.unpbook.com`). In Chapter 11, we will discuss the functions that convert between hostnames and IP addresses, and between service names and ports. We purposely put off the discussion of these functions and continue using IP addresses and port numbers so we know exactly what goes into the socket address structures that we must fill in and examine. This also avoids complicating our discussion of network programming with the details of yet another set of functions.