

## 8.6 UDP Echo Client: `dg_cli` Function

[Figure 8.8](#) shows the function `dg_cli`, which performs most of the client processing.

**Figure 8.8** `dg_cli` function: client processing loop.

*lib/dg\_cli.c*

```
1 #include      "unp.h"

2 void
3 dg_cli(FILE *fp, int sockfd, const SA *pservaddr, socklen_t servlen)
4 {
5     int      n;
6     char      sendline[MAXLINE], recvline[MAXLINE + 1];

7     while (Fgets(sendline, MAXLINE, fp) != NULL) {

8         Sendto(sockfd, sendline, strlen(sendline), 0, pservaddr, servlen);

9         n = Recvfrom(sockfd, recvline, MAXLINE, 0, NULL, NULL);

10        recvline[n] = 0;          /* null terminate */
11        Fputs(recvline, stdout);
12    }
13 }
```

**7–12** There are four steps in the client processing loop: read a line from standard input using `fgets`, send the line to the server using `sendto`, read back the server's echo using `recvfrom`, and print the echoed line to standard output using `fputs`.

Our client has not asked the kernel to assign an ephemeral port to its socket. (With a TCP client, we said the call to `connect` is where this takes place.) With a UDP socket, the first time the process calls `sendto`, if the socket has not yet had a local port bound to it, that is when an ephemeral port is chosen by the kernel for the socket. As with TCP, the client can call `bind` explicitly, but this is rarely done.

Notice that the call to `recvfrom` specifies a null pointer as the fifth and sixth arguments. This tells the kernel that we are not interested in knowing who sent the reply. There is a risk that any process, on either the same host or some other host, can send a datagram to the client's IP address and port, and that datagram will be read by the client, who will think it is the server's reply. We will address this in [Section 8.8](#).

As with the server function `dg_echo`, the client function `dg_cli` is protocol-independent, but the client `main` function is protocol-dependent. The `main` function allocates and initializes a socket address structure of some protocol type and then passes a pointer to this structure, along with its size, to `dg_cli`.