```
  printf("\nEnter a (hex) address: ");
  scanf("%x", &addr);
  printf("Enter number of bytes to view: ");
  scanf("%d", &n);

  printf("\n");
  printf(" Address              Bytes                Characters\n");
  printf(" -------  ---------------------------  ----------\n");

  ptr = (BYTE *) addr;
  for (; n > 0; n -= 10) {
    printf("%8X  ", (unsigned int) ptr);
    for (i = 0; i < 10 && i < n; i++)
      printf("%.2X ", *(ptr + i));
    for (; i < 10; i++)
      printf("   ");
    printf(" ");
    for (i = 0;  i < 10 && i < n; i++) {
      BYTE ch = *(ptr + i);
      if (!isprint(ch))
        ch = '.';
      printf("%c", ch);
    }
    printf("\n");
    ptr += 10;
  }

  return 0;
}
```

The program is complicated somewhat by the possibility that the value of n isn't a multiple of 10, so there may be fewer than 10 bytes in the last group. Two of the `for` statements are controlled by the condition `i < 10 && i < n`. This condition causes the loops to execute 10 times or `n` times, whichever is smaller. There's also a `for` statement that compensates for any missing bytes in the last group by displaying three spaces for each missing byte. That way, the characters that follow the last group of bytes will align properly with the character groups on previous lines.

The `%X` conversion specifier used in this program is similar to `%x`, which was discussed in Section 7.1. The difference is that `%X` displays the hexadecimal digits A, B, C, D, E, and F as upper-case letters; `%x` displays them in lower case.

Here's what happened when I compiled the program using GCC and tested it on an x86 system running Linux:

```
Address of main function: 804847c
Address of addr variable: bff41154

Enter a (hex) address: 8048000
Enter number of bytes to view: 40

 Address                  Bytes                  Characters
 -------  -------------------------------------  ----------
 8048000  7F 45 4C 46 01 01 01 00 00 00  .ELF......
 804800A  00 00 00 00 00 00 02 00 03 00  ..........
 8048014  01 00 00 00 C0 83 04 08 34 00  ........4.
 804801E  00 00 C0 0A 00 00 00 00 00 00  ..........
```