msg contains the following lines:

```
Trust not him with your secrets, who, when left
alone in your room, turns over your papers.
                --Johann Kaspar Lavater (1741-1801)
```

To encrypt the msg file, saving the encrypted message in newmsg, we'd use the following command:

```
xor <msg >newmsg
```

newmsg will now contain these lines:

```
rTSUR HIR NOK QORN _IST UCETCRU, QNI, QNCH JC@R
GJIHC OH _IST TIIK, RSTHU IPCT _IST VGVCTU.
               --lINGHH mGUVGT jGPGRCT (1741-1801)
```

To recover the original message, we'd use the command

```
xor <newmsg
```

which will display it on the screen.

As the example shows, our program won't change some characters, including digits. XORing these characters with & would produce invisible control characters, which could cause problems with some operating systems. In Chapter 22, we'll see how to avoid problems when reading and writing files that contain control characters. Until then, we'll play it safe by using the `isprint` function to make sure that both the original character and the new (encrypted) character are printing characters (i.e., not control characters). If either character fails this test, we'll have the program write the original character instead of the new character.

`isprint` function ➤23.5

Here's the finished program, which is remarkably short:

xor.c

```
/* Performs XOR encryption */

#include <ctype.h>
#include <stdio.h>

#define KEY '&'

int main(void)
{
  int orig_char, new_char;

  while ((orig_char = getchar()) != EOF) {
    new_char = orig_char ^ KEY;
    if (isprint(orig_char) && isprint(new_char))
      putchar(new_char);
    else
      putchar(orig_char);
  }

  return 0;
}
```