Q: When I'm reading user input, how can I skip all characters left on the current input line?

A: One possibility is to write a small function that reads and ignores all characters up to (and including) the first new-line character:

```
void skip_line(void)
{
  while (getchar() != '\n')
    ;
}
```

Another possibility is to ask `scanf` to skip all characters up to the first new-line character:

```
scanf("%*[^\n]");    /* skips characters up to new-line */
```

`scanf` will read all characters up to the first new-line character, but not store them anywhere (the `*` indicates assignment suppression). The only problem with using `scanf` is that it leaves the new-line character unread, so you may have to discard it separately.

Whatever you do, don't call the `fflush` function:

```
fflush(stdin);    /* effect is undefined */
```

Although some implementations allow the use of `fflush` to "flush" unread input, it's not a good idea to assume that all do. `fflush` is designed to flush *output* streams; the C standard states that its effect on input streams is undefined.

Q: Why is it not a good idea to use `fread` and `fwrite` with text streams? [p. 571]

A: One difficulty is that, under some operating systems, the new-line character becomes a pair of characters when written to a text file (see Section 22.1 for details). We must take this expansion into account, or else we're likely to lose track of our data. For example, if we use `fwrite` to write blocks of 80 characters, some of the blocks may end up occupying more than 80 bytes in the file because of new-line characters that were expanded.

Q: Why are there two sets of file-positioning functions (`fseek`/`ftell` and `fsetpos`/`fgetpos`)? Wouldn't one set be enough? [p. 574]

A: `fseek` and `ftell` have been part of the C library for eons. They have one drawback, though: they assume that a file position will fit in a `long int` value. Since `long int` is typically a 32-bit type, this means that `fseek` and `ftell` may not work with files containing more than 2,147,483,647 bytes. In recognition of this problem, `fsetpos` and `fgetpos` were added to `<stdio.h>` when C89 was created. These functions aren't required to treat file positions as numbers, so they're not subject to the `long int` restriction. But don't assume that you have to use `fsetpos` and `fgetpos`; if your implementation supports a 64-bit `long int` type, `fseek` and `ftell` are fine even for very large files.