Table 22.16 scanf Examples (Group 3)

scanf *Call*	*Input*	*Variables*
`n = scanf("%i%i%i", &i, &j, &k);`	~~12•012•0x12~~␣	n: 3 i: 12 j: 10 k: 18
`n = scanf("%[0123456789]", str);`	~~123~~abc␣	n: 1 str: "123"
`n = scanf("%[0123456789]", str);`	abc123␣	n: 0 str: unchanged
`n = scanf("%[^0123456789]", str);`	~~abc~~123␣	n: 1 str: "abc"
`n = scanf("%*d%d%n", &i, &j);`	~~10•20•~~30␣	n: 1 i: 20 j: 5

Detecting End-of-File and Error Conditions

```
void clearerr(FILE *stream);
int feof(FILE *stream);
int ferror(FILE *stream);
```

If we ask a ...`scanf` function to read and store *n* data items, we expect its return value to be *n*. If the return value is less than *n*, something went wrong. There are three possibilities:

- ***End-of-file.*** The function encountered end-of-file before matching the format string completely.
- ***Read error.*** The function was unable to read characters from the stream.
- ***Matching failure.*** A data item was in the wrong format. For example, the function might have encountered a letter while searching for the first digit of an integer.

But how can we tell which kind of failure occurred? In many cases, it doesn't matter: something went wrong, and we've got to abandon the program. There may be times, however, when we'll need to pinpoint the reason for the failure.

Every stream has two indicators associated with it: an ***error indicator*** and an ***end-of-file indicator***. These indicators are cleared when the stream is opened. Not surprisingly, encountering end-of-file sets the end-of-file indicator, and a read error sets the error indicator. (The error indicator is also set when a write error occurs on an output stream.) A matching failure doesn't change either indicator.

clearerr

Once the error or end-of-file indicator is set, it remains in that state until it's explicitly cleared, perhaps by a call of the `clearerr` function. `clearerr` clears both the end-of-file and error indicators:

```
clearerr(fp);     /* clears eof and error indicators for fp */
```