## Input Functions

```
int fgetc(FILE *stream);
int getc(FILE *stream);
int getchar(void);
int ungetc(int c, FILE *stream);
```

getchar

`getchar` reads a character from the `stdin` stream:

```
ch = getchar();    /* reads a character from stdin */
```

fgetc
getc

`fgetc` and `getc` read a character from an arbitrary stream:

```
ch = fgetc(fp);    /* reads a character from fp */
ch = getc(fp);     /* reads a character from fp */
```

All three functions treat the character as an `unsigned char` value (which is then converted to `int` type before it's returned). As a result, they never return a negative value other than EOF.

The relationship between `getc` and `fgetc` is similar to that between `putc` and `fputc`. `getc` is usually implemented as a macro (as well as a function), while `fgetc` is implemented only as a function. `getchar` is normally a macro as well:

```
#define getchar() getc(stdin)
```

For reading characters from a file, programmers usually prefer `getc` over `fgetc`. Since `getc` is normally available in macro form, it tends to be faster. `fgetc` can be used as a backup if `getc` isn't appropriate. (The standard allows the `getc` macro to evaluate its argument more than once, which may be a problem.)

The `fgetc`, `getc`, and `getchar` functions behave the same if a problem occurs. At end-of-file, they set the stream's end-of-file indicator and return EOF. If a read error occurs, they set the stream's error indicator and return EOF. To differentiate between the two situations, we can call either `feof` or `ferror`.

One of the most common uses of `fgetc`, `getc`, and `getchar` is to read characters from a file, one by one, until end-of-file occurs. It's customary to use the following `while` loop for that purpose:

idiom

```
while ((ch = getc(fp)) != EOF) {
  ...
}
```

After reading a character from the file associated with `fp` and storing it in the variable `ch` (which must be of type `int`), the `while` test compares `ch` with EOF. If `ch` isn't equal to EOF, we're not at the end of the file yet, so the body of the loop is executed. If `ch` is equal to EOF, the loop terminates.