Selected Functions in the C String Library

Except where noted, these functions are declared in <string.h> (for C) or <cstring> (for C++)
The C String Library operates on "C-strings" - char arrays with a terminal null byte ('\0'). Do not confuse these facilities with the C++ std::string class facilities.

`size_t strlen(const char * s)`
Returns the number of characters in string s not counting the terminal null byte. The type size_t is typedef'd in the C library to a large integer type (usually unsigned); assigning the result to an int variable is normally acceptable, but you might need a cast to avoid a compiler warning.

`char * strcpy(char * dest, const char * src)`
Copies string src to string dest, including the trailing null byte, and returns dest.

`char * strncpy(char * dest, const char * src, size_t n)`
Copies exactly n characters of src to dest. If src is shorter than n, then appends null bytes to dest until n characters are copied. If src length is greater than or equal to n, then n characters are copied from src to dest and dest has no null byte appended.

`int strcmp(const char * s1, const char * s2)`
Lexicographically compares string s1 to string s2 and returns a value:
< 0 s1 is less than s2
$= 0$ s1 is equal to s2
> 0 s1 is greater than s2
Comparison stops when unequal characters are found or when a string terminates.
Note: This function is case-sensitive. The C or C++ Standard Library does not have a case-insensitive comparison function.

`char * strcat(char * dest, const char * src)`
Appends string src to string dest, including the trailing null byte, and returns dest.

`char * strncat(char * dest, const char * src, size_t n)`
Appends characters from src to dest until n characters are appended (in which case a null character is added) or until a null character is appended from src; thus, up to n+1 characters may be appended

`char * fgets(char * s, int n, FILE * stream)  /* declared in <stdio.h> */`
This function is useful for safely reading an entire line from the stream into the character array s. The size n should be the size of the array whose starting address is s. The function reads characters from the stream and stores them in the array until one of the following happens:

1. n-1 characters have been read and stored. A null byte is stored in the last cell of the array, and s is returned.
2. A newline character has been read and stored. A null byte is stored in the next cell of the array, and s is returned.
3. End-of-file was encountered after reading and storing at least one character. A null byte is stored in the next cell of the array, and s is returned.
4. End-of-file was encountered before reading and storing any characters. A null pointer is returned.
5. An I/O error occurs. A null pointer is returned, and the contents of s are undefined - use **feof** to determine whether the returned null pointer is due to end-of-file or an error condition.

Note: In C++, use the get_line function with a std::string and std::istream variable.

`int fputs(char * s, FILE * stream)  /* declared in <stdio.h> */`
This function simply outputs all of the characters in s to the stream, up to, but not including, the null byte. If an error occurs, EOF is returned.

Occasionally Useful Functions

`char * strchr(const char * s, int c)`
Searches string s for the first occurrence of character c and returns:
pointer to character, if found
null pointer, otherwise

`char * strrchr(const char * s, int c)`
Searches string s for the last occurrence of character c and returns:
pointer to character, if found
null pointer, otherwise

`int strncmp(const char * s1, const char * s2, size_t n)`
Lexicographically compares up to n characters of string s1 to string s2 and returns a value:
< 0 s1 is less than s2
$= 0$ s1 is equal to s2
> 0 s1 is greater than s2
Comparison stops when unequal characters are found, when a string terminates, or when n characters have been compared.

`char * strpbrk(const char * s, const char * set)`
Searches string s for the first occurrence of a character from string set and returns:
pointer to character, if found
null pointer, otherwise

`size_t strspn(const char * s, const char * set)`
Searches string s for the first occurrence of a character not in string set and returns the length of the longest initial segment of s that consists of characters from string set.

`size_t strcspn(const char * s, const char * set)`
Searches string s for the first occurrence of a character from string set and returns the length of the longest initial segment of s that consists of characters not in string set.

`char * strstr(const char * src,  const char * sub)`
Searches string src for the first occurrence of substring sub and returns:
pointer to first character of substring, if found
null pointer, otherwise

`char * strtok(char * str, const char * set)`
Repeated calls on this function allow the tokenizing of a string str in which the tokens are separated by characters from the string set. See the text or one of the references for a full description of the function and its usage.