Builtin awk variables:

ARGC ARGV : Both are analagous to C, but I'm not sure what they actually include...

ENVIRON : An array of environment variable values. The array is indexed by variable name, each element being the value of that variable. Thus, the environment variable HOME is ENVIRON["HOME"].

FILENAME : The name of the current input file. If no files are specified on the command line, the value of FILENAME is the null string.

FNR : The input record number in the current input file.

FS : The input field separator, a space by default.

IGNORECASE : The case-sensitivity flag for string comparisons and regular expression operations. If IGNORECASE has a non-zero value, then pattern matching and all string comparisons are done ignoring case.

NF : The number of fields in the current input record, number of columns

NR : The total number of input records seen so far, number of lines

OFMT : The default output format for the print statement, ".6g" by default. Accepts most C style printf formats if I'm not mistaken.

OFS : The output field separator, a space by default, when using print $1,$2. You can always specify your own just by using printf instead of print.

ORS : The output record separator, by default a newline.

RS : The input record separator (line separator), by default a newline. Don't really have a good example of why you would change this, but you can.

There are a few more, but again, I don't see why you'd want to use them.

Operators:

()  Use them for arithmetic grouping.

$[1-NF] Field reference. $0 prints the whole line.

++ -- Increment and decrement, both prefix and postfix a la C.

^ Exponentiation ("**" may also be used, but beware; it worked on my office machine, but it did NOT work on this laptop XUbuntu Gusty 7.10).

+ - !  Unary plus, unary minus, and logical negation.

* / % Multiplication, division, and modulus.

+ - Addition and subtraction.

space String concatenation.

< <= > >= != == The usual relational operators.

~ !~ Regular expression match, negated match.

in Array membership. I didn't talk about this, but associative arrays are some of the most powerful things for processing logfiles and the like.

&& || Logical "and", Logical "or" a la C
?: Bitwise operators, a la C. If you're not familiar with these, it's like an ultra compact \texttt{if/else} statement. \texttt{`expr1 ? expr2 : expr3'}. If \texttt{expr1} is true, the value of the expression is \texttt{expr2}; otherwise it is \texttt{expr3}. Only one of \texttt{expr2} and \texttt{expr3} is evaluated.

\texttt{=} \texttt{+=} \texttt{-=} \texttt{*=} \texttt{/=} \texttt{=} \texttt{=} \texttt{ Assignment. 'Nuff said.}

=========
Control Statements

\texttt{if} (condition) statement
\hspace{1em} [ \texttt{else statement} ]

\texttt{while} (condition) statement

\texttt{do} statement \texttt{while} (condition)

\texttt{for} (expr1; expr2; expr3) statement

\texttt{for} (var in array) statement

\texttt{break}

\texttt{continue}

\texttt{delete} array[index]

\texttt{delete} array

\texttt{exit} [ expression ]

=========
The built-in arithmetic functions are:

\texttt{atan2(y, x)} the arctangent of \texttt{y/x} in radians.

\texttt{cos(expr)} the cosine in radians.

\texttt{exp(expr)} the exponential \texttt{function} (\texttt{e ^ expr}).

\texttt{int(expr)} truncates to integer.

\texttt{log(expr)} the natural logarithm of \texttt{expr}.

\texttt{rand()} a random number between zero and one.

\texttt{sin(expr)} the sine in radians.

\texttt{sqrt(expr)} the square root \texttt{function}.

\texttt{srand([expr])} use \texttt{expr} as a new seed for the random number generator. If no \texttt{expr} is provided, the time of day is used. The \texttt{return} value is the previous seed for the random number generator.

=========
\texttt{awk} has the following built-in string functions:

\texttt{length([str])} returns the \texttt{length} of the string \texttt{str}. The \texttt{length} of \texttt{$0} is returned if no argument is supplied.

\texttt{match(str, regex)} returns the position in \texttt{str} where the regular expression \texttt{regex} occurs, or zero if \texttt{regex} is not present, and sets the values of \texttt{RSTART} and \texttt{RLENGTH}.

\texttt{split(str, arr [, regex])} splits the string \texttt{str} into the array \texttt{arr} on the regular expression \texttt{regex}, and returns the number of elements. If \texttt{regex} is omitted, \texttt{FS} is used instead. \texttt{regex} can be the null string, causing each character to be placed into its own array element. The array \texttt{arr} is cleared first.

\texttt{printf(fmt, expr-list)} prints \texttt{expr-list} according to \texttt{fmt}, and returns the resulting string.
substr(str, index [, len]) returns the len-character substring of str starting at index. If len is omitted, the rest of str is used.

tolower(str) returns a copy of the string str, with all the upper-case characters in str translated to their corresponding lower-case counterparts. Non-alphabetic characters are left unchanged.

toupper(str) returns a copy of the string str, with all the lower-case characters in str translated to their corresponding upper-case counterparts. Non-alphabetic characters are left unchanged.

system(cmd-line) Execute the command cmd-line, and return the exit status. If your operating system does not support system, calling it will generate a fatal error.

systime() returns the current time of day as the number of seconds since a particular epoch (Midnight, January 1, 1970 UTC, on POSIX systems).

==========
This is only a selected list, and you can see how long it is! As always, just play around with it until you get the hang of it.