```
#!/bin/awk

==========
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, its like an ulra compact **if**/**else** statement. `expr1 ? expr2 : expr3'. If expr1 is true, the value of the expression is expr2; otherwise it is expr3. Only one of expr2 and expr3 is evaluated.

= += -= *= /= %= ^= Assignment. 'Nuff said.

==========
Control Statements

if (condition) statement
[else statement]

while (condition) statement

do statement **while** (condition)

for (expr1; expr2; expr3) statement

for (var in array) statement
break
continue
delete array[index]
delete array
exit [expression]

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

atan2(y, x) the arctangent of y/x in radians.

cos(expr) the cosine in radians.

exp(expr) the exponential **function** (e ^ expr).

int(expr) truncates to integer.

log(expr) the natural logarithm of expr.

rand() a random number between zero and one.

sin(expr) the sine in radians.

sqrt(expr) the square root **function**.

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

==========
awk has the following built-in string functions:

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

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

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

sprintf(fmt, expr-list) prints expr-list according to 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.