= 2 rounds to determine the winner , a tournament of 32 teams requires log2 32 =
5 rounds , etc . In this case , for n players / teams where n is not a power of 2 , log2 n is rounded up since it is necessary to have at least one round in which not all remaining competitors play . For example , log2 6 is approximately 2 @.@ 585 , which rounds up to 3 , indicating that a tournament of 6 teams requires 3 rounds (either two teams sit out the first round , or one team sits out the second round) . The same number of rounds is also necessary to determine a clear winner in a Swiss @-@ system tournament .

= = = Photography = = =

In photography , exposure values are measured in terms of the binary logarithm of the amount of light reaching the film or sensor , in accordance with the Weber ? Fechner law describing a logarithmic response of the human visual system to light . A single stop of exposure is one unit on a base @-@ 2 logarithmic scale . More precisely , the exposure value of a photograph is defined as
<formula>
where N is the f @-@ number measuring the aperture of the lens during the exposure , and t is the number of seconds of exposure .
Binary logarithms (expressed as stops) are also used in densitometry , to express the dynamic range of light @-@ sensitive materials or digital sensors .

= = Calculation = =

= = = Conversion from other bases = = =

An easy way to calculate log2 n on calculators that do not have a log2 function is to use the natural logarithm (ln) or the common logarithm (log or log10) functions , which are found on most scientific calculators . The specific change of logarithm base formulae for this are :
<formula>
or approximately
<formula>

= = = Integer rounding = = =

The binary logarithm can be made into a function from integers and to integers by rounding it up or down . These two forms of integer binary logarithm are related by this formula :
<formula>
The definition can be extended by defining <formula> . Extended in this way , this function is related to the number of leading zeros of the 32 @-@ bit unsigned binary representation of x , nlz (x) .
<formula>
The integer binary logarithm can be interpreted as the zero @-@ based index of the most significant 1 bit in the input . In this sense it is the complement of the find first set operation , which finds the index of the least significant 1 bit . Many hardware platforms include support for finding the number of leading zeros , or equivalent operations , which can be used to quickly find the binary logarithm . The fls and flsl functions in the Linux kernel and in some versions of the libc software library also compute the binary logarithm (rounded up to an integer , plus one) .

= = = Iterative approximation = = =

For a general positive real number , the binary logarithm may be computed in two parts . First , one computes the integer part , <formula> (called the characteristic of the logarithm) . This reduces the problem to one where the argument of the logarithm is in a restricted range , the interval [1 @,@ 2) , simplifying the second step of computing the fractional part (the mantissa of the logarithm) . For any x > 0 , there exists a unique integer n such that 2n ? x < 2n + 1 , or equivalently 1 ? 2 ? nx < 2 . Now the integer part of the logarithm is simply n , and the fractional part is log2 (2 ? nx) . In other words :

<formula>

For normalized floating point numbers , the integer part is given by the floating point exponent , and for integers it can be determined by performing a count leading zeros operation .

The fractional part of the result is log2 y , and can be computed iteratively , using only elementary multiplication and division . The algorithm for computing the fractional part can be described in pseudocode as follows :