NAME

fmlexpr – evaluate arguments as an expression

SYNOPSIS

fmlexpr *arguments*

DESCRIPTION

The **fmlexpr** function evaluates its arguments as an expression. After evaluation, the result is written on the standard output. Terms of the expression must be separated by blanks. Characters special to FMLI must be escaped. Note that **30** is returned to indicate a zero value, rather than the null string. Strings containing blanks or other special characters should be quoted. Integer-valued arguments may be preceded by a unary minus sign. Internally, integers are treated as 32-bit, 2s complement numbers.

The operators and keywords are listed below. Characters that need to be escaped are preceded by \. The list is in order of increasing precedence, with equal precedence operators grouped within **{}** symbols.

USAGE

Expressions

expr **\|** *expr* Returns the first *expr* if it is neither **NULL** nor **0**, otherwise returns the second *expr*.

expr **\&** *expr* Returns the first *expr* if neither *expr* is **NULL** or **0**, otherwise returns **0**.

expr { **=**, **\>**, **\>=**, **\<**, **\<=**, **!=** } *expr*
Returns the result of an integer comparison if both arguments are integers, otherwise returns the result of a lexical comparison.

expr { **+**, – } *expr* Addition or subtraction of integer-valued arguments.

expr { ∗, **/**, **%** } *expr*
Multiplication, division, or remainder of the integer-valued arguments.

expr **:** *expr* The matching operator **:** compares the first argument with the second argument which must be a regular expression. Regular expression syntax is the same as that of **ed**(1), except that all patterns are ''anchored'' (that is, begin with ^) and, therefore, ^ is not a special character, in that context. Normally, the matching operator returns the number of bytes matched (**0** on failure). Alternatively, the **\(...\)** pattern symbols can be used to return a portion of the first argument.

EXAMPLES

1. Add 1 to the variable **a**:
   example% 'fmlexpr $a + 1 | set -l a'
2. For $a equal to either "/usr/abc/file" or just "file":
   example% fmlexpr $a : .∗/\(.∗\) \| $a

   returns the last segment of a path name (that is, *file*). Watch out for **/** alone as an argument: **fmlexpr** will take it as the division operator (see **NOTES** below).

3. A better representation of example 2.
   example% fmlexpr //$a : .*/\(.*\)

   The addition of the **//** characters eliminates any ambiguity about the division operator (because it makes it impossible for the left-hand expression to be interpreted as the division operator), and simplifies the whole expression.
4. Return the number of characters in **$VAR.**
   example% fmlexpr $VAR : .*

EXIT CODES

As a side effect of expression evaluation, **fmlexpr** returns the following exit values:

0	if the expression is neither **NULL** nor **0** (that is, TRUE)
1	if the expression *is* **NULL** or **0** (that is, FALSE)
2	for invalid expressions (that is, FALSE).

SEE ALSO

ed(1), **expr**(1), **set**(1F), **sh**(1)

DIAGNOSTICS

syntax error for operator/operand errors
non-numeric argument
if arithmetic is attempted on such a string

In the case of syntax errors and non-numeric arguments, an error message will be printed at the current cursor position. Use **refresh** to redraw the screen.

NOTES

After argument processing by FMLI, **fmlexpr** cannot tell the difference between an operator and an operand except by the value. If **$a** is an **=**, the command:

example% fmlexpr $a = =

looks like:

example% fmlexpr = = =

as the arguments are passed to **fmlexpr** (and they will all be taken as the **=** operator). The following works, and returns TRUE:

example% fmlexpr X$a = X=