NAME dc – desk calculator

SYNOPSIS **dc** [*filename*]

AVAILABILITY SUNWesu

DESCRIPTION

dc is an arbitrary precision arithmetic package. Ordinarily it operates on decimal integers, but one may specify an input base, output base, and a number of fractional digits to be maintained. The overall structure of **dc** is a stacking (reverse Polish) calculator. If an argument is given, input is taken from that file until its end, then from the standard input.

bc is a preprocessor for **dc** that provides infix notation and a C-like syntax that implements functions. **bc** also provides reasonable control structures for programs. See **bc**(1).

USAGE

The following constructions are recognized:

number — The value of the number is pushed on the stack. A number is an unbroken string of the digits 0–9. It may be preceded by an underscore (_) to input a negative number. Numbers may contain decimal points.

+ – / * % ^ — The top two values on the stack are added (**+**), subtracted (**–**), multiplied (*****), divided (**/**), remaindered (**%**), or exponentiated (**^**). The two entries are popped off the stack; the result is pushed on the stack in their place. Any fractional part of an exponent is ignored.

s*x* — The top of the stack is popped and stored into a register named *x*, where *x* may be any character. If the **s** is capitalized, *x* is treated as a stack and the value is pushed on it.

l*x* — The value in register *x* is pushed on the stack. The register *x* is not altered. All registers start with zero value. If the **l** is capitalized, register *x* is treated as a stack and its top value is popped onto the main stack.

d — The top value on the stack is duplicated.

p — The top value on the stack is printed. The top value remains unchanged.

P — Interprets the top of the stack as an ASCII string, removes it, and prints it.

f — All values on the stack are printed.

q — Exits the program. If executing a string, the recursion level is popped by two.

Q — Exits the program. The top value on the stack is popped and the string execution level is popped by that value.

x — Treats the top element of the stack as a character string and executes it as a string of **dc** commands.

X — Replaces the number on the top of the stack with its scale factor.

[...] — Puts the bracketed ASCII string onto the top of the stack.

<*x* >*x* =*x* The top two elements of the stack are popped and compared. Register .Ix is evaluated if they obey the stated relation.

v Replaces the top element on the stack by its square root. Any existing fractional part of the argument is taken into account, but otherwise the scale factor is ignored.

! Interprets the rest of the line as a shell command.

c All values on the stack are popped.

i The top value on the stack is popped and used as the number radix for further input.

I Pushes the input base on the top of the stack.

o The top value on the stack is popped and used as the number radix for further output.

O Pushes the output base on the top of the stack.

k The top of the stack is popped, and that value is used as a non-negative scale factor: the appropriate number of places are printed on output, and maintained during multiplication, division, and exponentiation. The interaction of scale factor, input base, and output base will be reasonable if all are changed together.

K Pushes the current scale factor on the top of the stack.

z The stack level is pushed onto the stack.

Z Replaces the number on the top of the stack with its length.

? A line of input is taken from the input source (usually the terminal) and executed.

Y Displays **dc** debugging information.

; : are used by **bc**(1) for array operations.

EXAMPLES

This example prints the first ten values of n!:

```
[la1+dsa*pla10>y]sy
0sa1
lyx
```

SEE ALSO

bc(1)

DIAGNOSTICS

x **is unimplemented**	*x* is an octal number.
out of space	The free list is exhausted (too many digits).
out of stack space	Too many pushes onto the stack (stack overflow).
empty stack	Too many pops from the stack (stack underflow).
nesting depth	Too many levels of nested execution.
divide by 0	Division by zero.

sqrt of neg number	Square root of a negative number is not defined (no imaginary numbers).
exp not an integer	**dc** only processes integer exponentiation.
exp too big	The largest exponent allowed is 999.
input base is too large	The input base x: 2<= x <= 16.
input base is too small	The input base x: 2<= x <= 16.
output base is too large	The output base must be no larger than **BC_BASE_MAX**.
invalid scale factor	Scale factor cannot be less than 1.
scale factor is too large	A scale factor cannot be larger than **BC_SCALE_MAX**.
symbol table overflow	Too many variables have been specified.
invalid index	Index cannot be less than 1.
index is too large	An index cannot be larger than **BC_DIM_MAX**.