NAME

calc − arbitrary precision calculator

SYNOPSIS

calc [**−c**] [**−C**] [**−d**]
[**-D calc_debug[:resource_debug[:user_debug]]**]
[**−e**] [**−h**] [**−i**] [**−m mode**] [**−O**]
[**−p**] [**−q**] [**−s**] [**−u**] [**−v**] [**calc_cmd . . .**]

#!c:/progra~1/Calc/bin/calc [other_flags ...] **−f**

DESCRIPTION

CALC OPTIONS

−c Continue reading command lines even after an execution error has caused the abandonment of a line.

For example:

calc read many_errors.cal

will cause **calc** to abort on the first error, whereas:

calc -c read many_errors.cal

will cause **calc** to try to process each line being read despite the errors that it encounters.

By default, calc startup resource files are silently ignored if not found. This flag will report missing startup resource files unless **−d** is also given.

−C Permit the execution of custom builtin functions. Without this flag, calling the custom() builtin function will simply generate an error.

Use of this flag may cause **calc** to execute functions that are non-standard and that are not portable. Custom builtin functions are disabled by default for this reason.

−d Disable the printing of the opening title. The printing of resource file debug and informational messages is also disabled as if **config("resource_debug", 0)** had been executed.

For example:

calc "read qtime; qtime(2)"

will output something like:

qtime(utc_hr_offset) defined
It’s nearly ten past six.

whereas:

calc -d "read qtime; qtime(2)"

will just say:

It’s nearly ten past six.

This flag disables the reporting of missing calc startup resource files.

-D calc_debug[:resource_debug[:user_debug]]
Force the initial value of config("calc_debug"), config("resource_debug") and config("user_debug").

The : separated strings are interpreted as signed 32 bit integers. After an optional leading sign a leading zero indicates octal conversion, and a leading ‘‘0x’’ or ‘‘0X’’ hexadecimal conversion. Otherwise, decimal conversion is assumed.

By default, *calc_debug* is 0, *resource_debug* is 3 and *user_debug* is 0.

For more information use the following **calc** command:

help config

−e Ignore any environment variables on startup. The getenv() builtin will still return values, however.

−f This flag is required when using calc in **shell script mode**. It must be at the end of the initial **#!** line of the script.

This flag is normally only at the end of a calc shell script. If the first line of an executable file begins **#!** followed by the absolute pathname of the **calc** program and the flag **−f** as in:

#!c:/progra~1/Calc/bin/calc [other_flags ...] **−f**

the rest of the file will be processed in **shell script mode**. See **SHELL SCRIPT MODE** section of this man page below for details.

The actual form of this flag is:

−f filename

On systems that treat an executable that begins with **#!** as a script, the path of the execurable is appended by the kernel as the final argument to the exec() system call. This is why the **−f** flag at the very end of the **#!** line.

It is possible use **−f filename** on the command line:

calc [other_flags ...] **−f** filename

This will cause calc to process lines in **filename** in **shell script mode**.

Use of **−f** implies **−s**. In addition, **−d** and **−p** are implied if **−i** is not given.

−h Print a help message. This option implies **−q**. This is equivalent to the **calc** command help help. The help facility is disabled unless the mode is 5 or 7. See **−m**.

−i Become interactive if possible. If *calc_cmd* args are given, **calc** by default, calc will execute them and exit. This flag causes **calc** to drop into interactive mode after the commands are executed. This flag will cause **calc** to drop into interactive mode after the commands are executed.

For example:

calc 2+5

will print the value 7 and exit whereas:

```
calc -i 2+5
```

will print the value 7 and prompt the user for more **calc** commands.

−m mode
This flag sets the permission mode of **calc**. It controls the ability for **calc** to open files and execute programs. *Mode* may be a number from 0 to 7.

The mode value is interpreted in a way similar to that of the **chmod**(1) octal mode:

0 do not open any file, do not execute progs
1 do not open any file
2 do not open files for reading, do not execute progs
3 do not open files for reading
4 do not open files for writing, do not execute progs
5 do not open files for writing
6 do not execute any program
7 allow everything (default mode)

If one wished to run **calc** from a privileged user, one might want to use **−m** 0 in an effort to make **calc** somewhat more secure.

Mode bits for reading and writing apply only on an open. Files already open are not effected. Thus if one wanted to use the **−m** 0 in an effort to make **calc** somewhat more secure, but still wanted to read and write a specific file, one might want to do in **sh**(1), **ksh**(1), **bash**(1)-like shells:

```
calc -m 0 3<a.file
```

Files presented to **calc** in this way are opened in an unknown mode. **Calc** will attempt to read or write them if directed.

If the mode disables opening of files for reading, then the startup resource files are disabled as if **−q** was given. The reading of key bindings is also disabled when the mode disables opening of files for reading.

−O Use the old classic defaults instead of the default configuration. This flag as the same effect as executing **config("all", "oldcfg")** at startup time.

NOTE: Older versions of calc used **−n** to setup a modified form of the default calc configuration. The **−n** flag currently does nothing. Use of the **−n** flag is now deprecated and may be used for something else in the future.

−p Pipe processing is enabled by use of **−p**. For example:

```
calc -p "2^21701-1" | fizzbin
```

In pipe mode, **calc** does not prompt, does not print leading tabs and does not print the initial header. The **−p** flag overrides **−i**.

−q Disable the reading of the startup scripts.

−s By default, all *calc_cmd* args are evaluated and executed. This flag will disable their evaluation and instead make them available as strings for the argv() builtin function.

−u Disable buffering of stdin and stdout.

−v Print the **calc** version number and exit.

CALC COMMAND LINE

With no *calc_cmd* arguments, **calc** operates interactively. If one or more arguments are given on the command line and **−s** is NOT given, then **calc** will read and execute them and either attempt to go interactive according as the **−i** flag was present or absent.

If **−s** is given, **calc** will not evaluate any *calc_cmd* arguments but instead make them available as strings to the argv() builtin function.

Sufficiently simple commands with no no characters like parentheses, brackets, semicolons, '*', which have special interpretations in UNIX shells may be entered, possibly with spaces, until the terminating newline. For example:

```
calc 23 + 47
```

should respond with display of 70, but

```
calc 23 * 47
```

may fail. Such cases can usually be made to work as expected by enclosing the command between single marks as in:

```
calc "23 * 47"
```

and

```
calc "print sqrt(2), exp(1)"
```

If '!' is to be used to indicate the factorial function, for shells like **csh***(1)* for which '!' followed by a non-space character is used for history substitution, it may be necessary to include a space or use a backslash to escape the special meaning of '!'. For example, the command:

```
print 27!^2
```

may have to be replaced by:

```
print 27! ^2        or        print 27^2
```

CALC STARTUP FILES

Normally on startup, if the environment variable **$CALCRC** is undefined and **calc** is invoked without the **−q** flag, or if **$CALCRC** is defined and calc is invoked with **−e**, **calc** looks for a file "startup" in the calc resource directory **.calcrc** in the user's home directory, and **.calcinit in the current directory.** If one or more of these are found, they are read in succession as **calc** scripts and their commands executed. When defined, **$CALCRC** is to contain a ':' separated list of names of files, and if calc is then invoked without either the **−q** or **−e** flags, these files are read in succession and their commands executed. No error condition is produced if a listed file is not found.

If the mode specified by **−m** disables opening of files for reading, then the reading of startup files is also disabled as if **−q** was given.

CALC FILE SEARCH PATH

If the environment variable **$CALCPATH** is undefined, or if it is defined and **calc** is invoked with the **−e** flag, when a file name not beginning with **/**, **˜** or **./**, is specified as in:

```
calc read myfile
```

calc searches in succession:

```
c:/progra˜1/Calc/lib/myfile
c:/progra˜1/Calc/lib/myfile.cal
c:/progra˜1/Calc/share/calc/calc/custom/myfile
c:/progra˜1/Calc/share/calc/calc/custom/myfile.cal
```

If the file is found, the search stops and the commands in the file are executed. It is an error if no readable file with the specified name is found. An alternative search path can be specified by defining **$CALCPATH** in the same way as PATH is defined, as a ':' separated list of directories, and then invoking **calc** without the **−e** flag.

Calc treats all open files, other than stdin, stdout and stderr as files available for reading and writing. One may present **calc** with an already open file using **sh**(1), **ksh**(1), **bash**(1)-like shells is to:

```
calc 3<open_file 4<open_file2
```

For more information use the following **calc** commands:

```
help help
help overview
help usage
help environment
help config
```

SHELL SCRIPT MODE

If the first line of an executable file begins **#!** followed by the absolute pathname of the **calc** program and the flag **−f** as in:

#!c:/progra˜1/Calc/bin/calc [other_flags ...] **−f**

the rest of the file will be processed in **shell script mode**. Note that **−f** must at the end of the intiial ''#!'' line. Any other optional **other_flags** must come before the **−f**.

In **shell script mode** the contents of the file are read and executed as if they were in a file being processed by a read command, except that a "command" beginning with '#' followed by whitespace and ending at the next newline is treated as a comment. Any optional **other_flags** will be parsed first followed by the later lines within the script itself.

In **shell script mode**, **−s** is always assumed. In addition, **−d** and **−p** are automatically set if **−i** is not given.

For example, if the file **/tmp/mersenne**:

```
#!c:/progra˜1/Calc/bin/calc −q −f
#
# mersenne - an example of a calc shell script file

/* parse args */
if (argv() != 1) {
    fprintf(files(2), "usage: %s exp\n", config("program"));
```

```
    abort "must give one exponent arg";
}

/* print the mersenne number */
print "2^": argv(0) : "-1 =", 2^eval(argv(0))-1;
```

is made an executable file by:

```
chmod +x /tmp/mersenne
```

then the command line:

```
/tmp/mersenne 127
```

will print:

```
2^127-1 = 170141183460469231731687303715884105727
```

Note that because **−s** is assumed in **shell script mode** and non-dashed args are made available as strings via the **argv**() builtin function. Therefore:

```
2^eval(argv(0))-1
```

will print the decimal value of 2^n-1 but

```
2^argv(0)-1
```

will not.

DATA TYPES

Fundamental builtin data types include integers, real numbers, rational numbers, complex numbers and strings.

By use of an object, one may define an arbitrarily complex data types. One may define how such objects behave a wide range of operations such as addition, subtraction, multiplication, division, negation, squaring, modulus, rounding, exponentiation, equality, comparison, printing and so on.

For more information use the following **calc** commands:

```
help types
help obj
show objfuncs
```

VARIABLES

Variables in *calc* are typeless. In other words, the fundamental type of a variable is determined by its content. Before a variable is assigned a value it has the value of zero.

The scope of a variable may be global, local to a file, or local to a procedure. Values may be grouped together in a matrix, or into a a list that permits stack and queue style operations.

For more information use the following **calc** commands:

```
help variable
help mat
help list
show globals
```

INPUT/OUTPUT

A leading ‘‘0x’’ implies a hexadecimal value, a leading ‘‘0b’’ implies a binary value, and a ‘‘0’’ followed by a digit implies an octal value. Complex numbers are indicated by a trailing ‘‘i’’ such as in ‘‘3+4i’’. Strings may be delimited by either a pair of single or double quotes. By default, *calc* prints values as if they were floating point numbers. One may change the default to print values in a number of modes including fractions, integers and exponentials.

A number of stdio-like file I/O operations are provided. One may open, read, write, seek and close files. Filenames are subject to ‘‘ ’’ expansion to home directories in a way similar to that of the Korn or C-Shell.

For example:

```
~/.calcrc
~chongo/lib/fft_multiply.cal
```

For more information use the following **calc** command:

```
help file
```

CALC LANGUAGE

The *calc* language is a C-like language. The language includes commands such as variable declarations, expressions, tests, labels, loops, file operations, function calls. These commands are very similar to their counterparts in C.

The language also include a number of commands particular to *calc* itself. These include commands such as function definition, help, reading in resource files, dump files to a file, error notification, configuration control and status.

For more information use the following **calc** command:

```
help command
help statement
help expression
help operator
help config
```

FILES

c:/progra~1/Calc/bin/calc
: calc binary

/lib/calc/cscript/*
: calc shell scripts

c:/progra~1/Calc/lib/*.cal
: calc standard resource files

c:/progra~1/Calc/lib/help/*
: help files

c:/progra~1/Calc/lib/bindings
: non-GNU-readline command line editor bindings

c:/progra~1/Calc/include/calc/*.h
: include files for C interface use

c:/progra~1/Calc/lib/libcalc.a
: calc binary link library

c:/progra~1/Calc/lib/libcustcalc.a
custom binary link library

c:/progra~1/Calc/share/calc/calc/custom/*.cal
custom resource files

c:/progra~1/Calc/share/calc/help/custhelp/*
custom help files

ENVIRONMENT

CALCPATH
A :-separated list of directories used to search for calc resource filenames that do not begin with /, ./ or ~.

Default value: .;./cal;~/.cal;c:/progra~1/Calc/share/calc;c:/progra~1/Calc/share/calc/calc/custom

CALCRC
On startup (unless −h or −q was given on the command line), **calc** searches for files along this :-separated environment variable.

Default value: c:/progra~1/Calc/share/calc/startup;~/.calcrc;./.calcinit

CALCBINDINGS
On startup (unless *−h* or *−q* was given on the command line, or *−m* disallows opening files for reading), **calc** reads key bindings from the filename specified by this environment variable. The key binding file is searched for along the $CALCPATH list of directories.

Default value: binding

This variable is not used if calc was compiled with GNU-readline support. In that case, the standard readline mechanisms (see readline(3)) are used.

CREDIT

The main chunk of **calc** was written by David I. Bell.

The **calc** primary mirror, calc mailing list and calc bug report processing is performed by Landon Curt Noll.

Landon Curt Noll maintains the master reference source, performs release control functions as well as other calc maintenance functions.

Thanks for suggestions and encouragement from Peter Miller, Neil Justusson, and Landon Noll.

Thanks to Stephen Rothwell for writing the original version of hist.c which is used to do the command line editing.

Thanks to Ernest W. Bowen for supplying many improvements in accuracy and generality for some numeric functions. Much of this was in terms of actual code which I gratefully accepted. Ernest also supplied the original text for many of the help files.

Portions of this program are derived from an earlier set of public domain arbitrarily precision routines which was posted to the net around 1984. By now, there is almost no recognizable code left from that original source.

COPYING / CALC GNU LESSER GENERAL PUBLIC LICENSE

Calc is open software, and is covered under version 2.1 of the GNU Lesser General Public License. You are welcome to change it and/or distribute copies of it under certain conditions. The calc commands:

help copyright
help copying
help copying-lgpl

should display the contents of the COPYING and COPYING-LGPL files. Those files contain information about the calc's GNU Lesser General Public License, and in particular the conditions under which you are allowed to change it and/or distribute copies of it.

You should have received a copy of the version 2.1 of the GNU Lesser General Public License. If you do not have these files, write to:

Free Software Foundation, Inc.
59 Temple Place
Suite 330
Boston, MA 02111-1307
USA

Calc is copyrighted in several different ways. These ways include:

Copyright (C) year David I. Bell
Copyright (C) year David I. Bell and Landon Curt Noll
Copyright (C) year David I. Bell and Ernest Bowen
Copyright (C) year David I. Bell, Landon Curt Noll and Ernest Bowen
Copyright (C) year Landon Curt Noll
Copyright (C) year Ernest Bowen and Landon Curt Noll
Copyright (C) year Ernest Bowen

This man page is:

Copyright (C) 1999 Landon Curt Noll

and is covered under version 2.1 GNU Lesser General Public License.

CALC MAILING LIST / CALC UPDATES / ENHANCEMENTS

To contribute comments, suggestions, enhancements and interesting **calc** resource files, and shell scripts please join the low volume calc mailing list.

To join the low volume calc mailing list, send EMail to:

calc-tester-request at asthe dot com

Your subject must contain the words:

calc mailing list subscription

You may have additional words in your subject line.

Your message body must contain:

subscribe calc-tester address
end
name your_full_name

where **address** s your EMail address and **your_full_name** is your full name. Feel free to follow the **name** line with additional EMail text as desired.

BUG REPORTS / BUG FIXES

Send bug reports and bug fixes to:

calc-bugs at asthe dot com

[[NOTE: Replace 'at' with @, 'dot' is with . and remove the spaces]]
[[NOTE: The EMail address uses 'asthe' and the web site URL uses 'isthe']]

Your subject must contain the words:

calc bug report

You may have additional words in your subject line.

See the *BUGS* source file or use the *calc* command:

help bugs

for more information about bug reporting.

CALC WEB SITE

Landon Noll maintains the the **calc** web site is located at:

www.isthe.com/chongo/tech/comp/calc/

Share and Enjoy! :−)