NAME

rcc − ANSI/ISO C compiler lexer, parser and code-generator component

SYNOPSIS

rcc −target=*name* [*option*]. . . [*infile* [*outfile*]]

Since **rcc** does not assume any default target, the **−target=***name* setting is mandatory, although it may be preceded by other options.

Standard input and output are assumed if the files are not supplied, or if their names are given as −.

DESCRIPTION

rcc is the compiler lexer, parser and code-generator component for the **lcc**(1) compiler system. **rcc** takes input from the C preprocessor, lexes and parses it, and generates assembly code for a requested target architecture and operating system. **lcc**(1) then normally invokes the native assembler and loader to produce the final executable file.

rcc is not normally invoked by humans, and indeed, is normally not found in the default **PATH**, but during compiler development work, it has sometimes been useful to do so, so these manual pages were written to document it.

OPTIONS

rcc interprets the options described in this section; unrecognized options are diagnosed as errors.

GNU/POSIX style **−−option** syntax is recognized as equivalent to **−option**.

−A Warn about declarations and casts of function types without prototypes, assignments between pointers to **int**s and pointers to **enum**s, and conversions from pointers to smaller integral types.

A second **−A** warns about unrecognized control lines, nonANSI/ISO language extensions and source characters in literals, unreferenced variables and static functions, declaring arrays of incomplete types, and exceeding *some* ANSI/ISO environmental limits, such as more than 257 cases in switches. It also arranges for duplicate global definitions in separately compiled files to cause loader errors.

−a*[input-profile-file]*
Read the specified profile file, or *prof.out* if the filename is omitted, from a previous execution and use the data therein to compute reference counts (see **−b**).

rcc assigns the most frequently-referenced scalar parameters and locals to registers whenever possible. For each block, explicit register declarations are obeyed first; remaining registers are assigned to automatic locals if they are ‘referenced’ at least 3 times. Each top-level occurrence of an identifier counts as 1 reference. Occurrences in a loop, either of the **then**/**else** arms of an **if** statement, or a **case** in a **switch** statement each count, respectively, as 10, 1/2, or 1/10 references. These values are adjusted accordingly for nested control structures.

−asdl Change the interface record (IR) function pointers to new ones prefixed by *asdl_*. [??? What is the purpose of this ???]

−b Produce code that counts the number of times each expression is executed. If loading takes place, arrange for a *prof.out* file to be written when the object program terminates. A listing annotated with execution counts can then be generated with **bprint**(1). **−C** is similar, but counts only the number of function calls.

−C Produce code to count the number of function calls. See also **−b**.

−copyright
Display copyright information on the standard output, and exit immediately with a success status code.

This option may be abbreviated to any unique prefix at least as long as **−co**.

−d Turn on debugging output.

−errout=*outfile*
Redirect the standard error unit to the specified output file. Execution terminates immediately with a failure status code if this action fails.

−e*n* Set the maximum number of error messages to *n*.

−g Produce additional symbol table information for the local debuggers.

−g*[n[,x]]*
Set the debugging level to *n* and emit source code as comments into the generated assembly code; *x* must be the assembly language comment character. If *n* is omitted, it defaults to 1, which is similar to **−g**. Omitting **,***x* just sets the debugging level to *n*.

−html When used with **−target=symbolic**, this option causes the text rendition to be emitted as strictly grammar-conformant HTML, complete with hypertext links for cross references in the code!

−left_to_right=1
−left_to_right=0
Set the interface record (IR) flag *left_to_right*. This controls function argument evaluation order, which is unspecified by ANSI/ISO Standard C; it is normally architecture dependent. [Reference: **lcc** book, p. 88]

−little_endian=1
−little_endian=0
Set the interface record (IR) flag *little_endian*: nonzero for little-endian addressing, or zero, for big-endian addressing. This value is normally architecture dependent, and may also be operating-system dependent on those CPU architectures that support both addressing forms. [Reference: **lcc** book, p. 87]

−*metricname=size,align,outofline*
Initialize the named metric with the three numeric parameters. The metric is one of **charmetric**, **doublemetric**, **floatmetric**, **intmetric**, **longdoublemetric**, **longlongmetric**, **longmetric**, **ptrmetric**, **shortmetric**, or **structmetric**. [Reference: **lcc** book, pp. 78--79]

−mulops_calls=1
−mulops_calls=0
Set the interface record (IR) flag *mulops_calls*: nonzero if multiply, divide, and remainder are done by library calls, or zero, if they are done in hardware. [Reference: **lcc** book, p. 87]

−n Produce code that tests for dereferencing zero pointers. The code reports the offending file and line number and calls **abort**(3).

−nvalidate[,check]
[??? What does this do ???]

−P Writes declarations for all defined globals on standard error. Function declarations include prototypes; editing this output can simplify conversion to ANSI/ISO C. This output may not correspond to the input when there are several **typedef**s for the same type.

−s *f* Generate jump tables for switches whose density is at least *f*, a floating-point constant between zero and one. The default is 0.5.

This is the same as **lcc**(1)'s **−d** *f* option.

−t*name*
−t Produce code to print the name of the function, an activation number, and the name and value of each argument at function entry. At function exit, produce code to print the name of the function, the activation number, and the return value. By default, *printf* does the printing; if *name* appears, it does. For null **char*** values, "(null)" is printed.

−target=*architecture/os*
Generate assembly code for the specified architecture and operating system. This switch is *mandatory*, since **rcc** does not assume any default target.

The supported *architecture*/*os* combinations may include

alpha/linux	Compaq/DEC Alpha, GNU/Linux
alpha/osf	Compaq/DEC Alpha, OSF/1 3.2, 4.x
mips/irix	big-endian MIPS Rx000, IRIX 5.2, 6.x
mips/linux	big-endian MIPS Rx000, GNU/Linux
mips/ultrix	little-endian MIPS Rx000, ULTRIX 4.3
null	no output
sparc/linux	Sun SPARC, GNU/Linux
sparc/solaris	Sun SPARC, Solaris 2.x
sparc/sun	Sun SPARC, SunOS 4.x
symbolic	text rendition of the generated code
symbolic/osf	text rendition of the generated code for Compaq/DEC OSF/1
symbolic/irix	text rendition of the generated code for SGI IRIX
x86/freebsd	Intel x86, FreeBSD
x86/linux	Intel x86, GNU/Linux
x86/solaris	Intel x86, Sun Solaris 2.x
x86/win32	Intel x86, Windows NT 4.0/Windows 95/98/2000

Additional combinations that may be supported in the future, if code-generation support is completed, include

ia64/freebsd	HP/Intel IA-64, FreeBSD
ia64/linux	HP/Intel IA-64, GNU/Linux
ia64/win32	HP/Intel IA-64, Windows NT 4.0/Windows 95/98/2000
ia64/win64	HP/Intel IA-64, Windows-64
parisc/hpux	HP PA-RISC, HP-UX 10.x, 11.x
parisc/linux	HP PA-RISC, GNU/Linux
ppc/aix	PowerPC, IBM AIX 4.x
ppc/linux	PowerPC, GNU/Linux
ppc/macosx	PowerPC, Mac OS X (Darwin, Rhapsody)

−unsigned_char=1
−unsigned_char=0

Make plain **char** an unsigned (1, or nonzero) or signed (0) type.

By default, **char** is signed on all platforms on which **lcc** runs. Note that this may differ from the choice made by other C compilers on the same platform.

−v Print the program name and RCS id string.

−version

Display version information on the standard output, and exit immediately with a success status code.

This option may be abbreviated to any unique prefix at least as long as **−ve**.

−w Suppress warning diagnostics, such as those announcing unreferenced statics, locals, and parameters. The line *#pragma ref id* simulates a reference to the variable *id*.

−wants_argb=1
−wants_argb=0

Set the interface record (IR) flag, *wants_argb*. When nonzero, the front end emits *ARGB* nodes to pass structure arguments; when zero, it uses simpler operations. [Reference: **lcc** book, p. 88]

−wants_callb=1
−wants_callb=0

Set the interface record (IR) flag *wants_callb*: nonzero if the front end should emit *CALLB* nodes to invoke functions that return structures, or zero for simpler code. [Reference: **lcc** book, p. 88]

−wants_dag=1
−wants_dag=0
Set the interface record (IR) flag *wants_dag*: nonzero if the front end should pass dags to the back end, or zero if the front end should undag all nodes with reference counts exceeding one. [Reference: **lcc** book, p. 89]

−wchar_t=unsigned_char
−wchar_t=unsigned_short
−wchar_t=unsigned_int
Makes wide characters the type indicated.

By default, for **lcc**, wide characters are **unsigned short int**, and **wchar_t** is a typedef defined in <*stddef.h*>. The definition for **wchar_t** in <*stddef.h*> changes according to the setting of this option. For that reason, care should be taken to ensure that the same value of this option is used if preprocessing is done separately from compilation.

−x Turn on some additional cross-referencing. [??? What does this do ???]

ENVIRONMENT VARIABLES

rcc does not use any environment variables.

FILES

rcc opens only those files explicitly given on its command line.

SEE ALSO

as(1), **bprint**(1), **c89**(1), **cc**(1), **collect2**(1), **cpp**(1), **gas**(1), **gcc**(1), **gprof**(1), **lcc**(1), **lcc-cpp**(1), **ld**(1), **pgcc**(1), **prof**(1).

C. W. Fraser and D. R. Hanson, *A Retargetable C Compiler: Design and Implementation*, Addison-Wesley, 1995. ISBN 0-8053-1670-1.

The World-Wide Web page at `http://www.cs.princeton.edu/software/lcc/`.

S. P. Harbison and G. L. Steele, Jr., *C: A Reference Manual*, 4th ed., Prentice-Hall, 1995.

B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, 2nd ed., Prentice-Hall, 1988.

American National Standards Inst., *American National Standard for Information Systems—Programming Language—C*, ANSI X3.159-1989, New York, 1990.

International Organization for Standardization, *ISO/IEC 9899:1990: Programming languages — C*, Geneva, Switzerland, 1990.

BUGS

Mail bug reports along with the shortest preprocessed program that exposes them, and the details reported by **rcc**'s **−v** option, to `lcc-bugs@princeton.edu`. The World-Wide Web page at URL `http://www.cs.princeton.edu/software/lcc/` includes detailed instructions for reporting bugs.