NAME

lcc − ANSI C compiler

SYNOPSIS

lcc [*option* | *file*]...

DESCRIPTION

lcc is an ANSI C compiler for a variety of platforms.

Arguments whose names end with ‘.c’ (plus ‘.C’ under Windows) are taken to be C source programs; they are preprocessed, compiled, and each object program is left on the file whose name is that of the source with ‘.o’ (UNIX) or ‘.obj’ (Windows) substituted for the extension. Arguments whose names end with ‘.i’ are treated similarly, except they are not preprocessed. In the same way, arguments ending with ‘.s’ (plus ‘.S’, ‘.asm’, and ‘.ASM’, under Windows) are taken to be assembly source programs and are assembled, producing an object file. If there are no arguments, *lcc* summarizes its options on the standard error.

lcc deletes an object file if and only if exactly one source file is mentioned and no other file (source, object, library) or **−l** option is mentioned.

If the environment variable **LCCINPUTS** is set, *lcc* assumes it gives a semicolon- or colon-separated list of directories in which to look for source and object files whose names do not begin with ‘/’. These directories are also added to the list of directories searched for libraries. If **LCCINPUTS** is defined, it must contain ‘.’ in order for the current directory to be searched for input files.

lcc uses ANSI standard header files (see ‘FILES’ below). Include files not found in the ANSI header files are taken from the normal default include areas, which usually includes **/usr/include**. Under Windows, if the environment variable **include** is defined, it gives a semicolon-separated list of directories in which to search for header files.

lcc interprets the following options; unrecognized options are taken as loader options (see *ld*(1)) unless **−c**, **−S**, or **−E** precedes them. Except for **−l**, all options are processed before any of the files and apply to all of the files. Applicable options are passed to each compilation phase in the order given.

−c Suppress the loading phase of the compilation, and force an object file to be produced even if only one program is compiled.

−g Produce additional symbol table information for the local debuggers. *lcc* warns when **−g** is unsupported.

−Wf−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*.

−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*.

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

−A Warns about declarations and casts of function types without prototypes, assignments between pointers to ints and pointers to enums, and conversions from pointers to smaller integral types. A second **−A** warns about unrecognized control lines, nonANSI language extensions and source characters in literals, unreferenced variables and static functions, declaring arrays of incomplete types, and exceeding *some* ANSI environmental limits, like more than 257 cases in switches. It also arranges for duplicate global definitions in separately compiled files to cause loader errors.

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

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

−O is ignored.

−S Compile the named C programs, and leave the assembler-language output on corresponding files suffixed '.s' or '.asm'.

−E Run only the preprocessor on the named C programs and unsuffixed file arguments, and send the result to the standard output.

−o *output*
Name the output file *output*. If **−c** or **−S** is specified and there is exactly one source file, this option names the object or assembly file, respectively. Otherwise, this option names the final executable file generated by the loader, and 'a.out' (UNIX) or 'a.exe' (Windows) is left undisturbed. *lcc* warns if **−o** and **−c** or **−S** are given with more than one source file and ignores the **−o** option.

−D*name*=*def*
Define the *name* to the preprocessor, as if by '#define'. If =*def* is omitted, the name is defined as "1".

−U*name*
Remove any initial definition of *name*.

−I*dir* '#include' files whose names do not begin with '/' are always sought first in the directory of the *file* arguments, then in directories named in **−I** options, then in directories on a standard list.

−N Do not search *any* of the standard directories for '#include' files. Only those directories specified by subsequent explicit **−I** options will be searched, in the order given.

−B*str* Use the compiler *str***rcc** instead of the default version. Note that *str* often requires a trailing slash. On Sparcs only, **−Bstatic** and **−Bdynamic** are passed to the loader; see *ld*(1).

−Wo−lccdir=*dir*
Find the preprocessor, compiler proper, and include directory in the directory *dir/* or *dir\*. If the environment variable **LCCDIR** is defined, it gives this directory. *lcc* warns when this option is unsupported.

−Wf-unsigned_char=1
−Wf-unsigned_char=0
makes plain **char** an unsigned (1) or signed (0) type; by default, **char** is signed.

−Wf−wchar_t=unsigned_char
−Wf−wchar_t=unsigned_short
−Wf−wchar_t=unsigned_int
Makes wide characters the type indicated; by default, wide characters are unsigned short ints, and **wchar_t** is a typedef for unsigned short defined in stddef.h. The definition for **wchar_t** in stddef.h must correspond to the type specified.

−v Print commands as they are executed; some of the executed programs are directed to print their version numbers. More than one occurrence of **−v** causes the commands to be printed, but *not* executed.

−help or **−?**
Print a message on the standard error summarizing *lcc*'s options and giving the values of the environment variables **LCCINPUTS** and **LCCDIR**, if they are defined. Under Windows, the values of **include** and **lib** are also given, if they are defined.

−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). *lcc* warns when **−b** is unsupported. **−Wf−C** is similar, but counts only the number of function calls.

−p Produce code that counts the number of times each function is called. If loading takes place, replace the standard startup function by one that automatically calls *monitor*(3) at the start and arranges to write a **mon.out** file when the object program terminates normally. An execution profile can then be generated with *prof*(1). *lcc* warns when **−p** is unsupported.

−pg Causes the compiler to produce counting code like **−p**, but invokes a run-time recording mechanism that keeps more extensive statistics and produces a **gmon.out** file at normal termination. Also, a profiling library is searched, in lieu of the standard C library. An execution profile can then be generated with *gprof*(1). *lcc* warns when **−pg** is unsupported.

−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** *name* is accepted, but ignored.

−tempdir=*dir*
Store temporary files in the directory *dir/* or *dir*. The default is usually **/tmp**.

−W*xarg*
pass argument *arg* to the program indicated by *x*; *x* can be one of **p**, **f**, **a**, or **l**, which refer, respectively, to the preprocessor, the compiler proper, the assembler, and the loader. *arg* is passed as given; if a **−** is expected, it must be given explicitly. **−Wo***arg* specifies a system-specific option, *arg*.

Other arguments are taken to be either loader option arguments, or C-compatible object programs, typically produced by an earlier *lcc* run, or perhaps libraries of C-compatible routines. Duplicate object files are ignored. These programs, together with the results of any compilations specified, are loaded (in the order given) to produce an executable program with name **a.out** (UNIX) or **a.exe** (Windows).

lcc 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. **−Wf−a** causes *lcc* to read a **prof.out** file from a previous execution and to use the data therein to compute reference counts (see **−b**).

lcc is a cross compiler; **−Wf−target=***target/os* causes *lcc* to generate code for *target* running the operating system denoted by *os*. The supported *target/os* combinations may include

alpha/osf	ALPHA, OSF 3.2
mips/irix	big-endian MIPS, IRIX 5.2
mips/ultrix	little-endian MIPS, ULTRIX 4.3
sparc/solaris	SPARC, Solaris 2.3
x86/win32	x86, Windows NT 4.0/Windows 95/98
x86/linux	x86, Linux
symbolic	text rendition of the generated code
null	no output

For **−Wf−target=symbolic**, the option **−Wf-html** causes the text rendition to be emitted as HTML.

LIMITATIONS

lcc accepts the C programming language as described in the ANSI standard. If *lcc* is used with the GNU C preprocessor, the **−Wp−trigraphs** option is required to enable trigraph sequences.

Plain int bit fields are signed. Bit fields are aligned like unsigned integers but are otherwise laid out as by most standard C compilers. Some compilers, such as the GNU C compiler, may choose other, incompatible layouts.

Likewise, calling conventions are intended to be compatible with the host C compiler, except possibly for passing and returning structures. Specifically, *lcc* passes and returns structures like host ANSI C compilers on most targets, but some older host C compilers use different conventions. Consequently, calls to/from such functions compiled with older C compilers may not work. Calling a function that returns a structure without declaring it as such violates the ANSI standard and may cause a fault.

FILES

The file names listed below are *typical*, but vary among installations; installation-dependent variants can be displayed by running *lcc* with the **−v** option.

file.{c,C}	input file
file.{s,asm}	assembly-language file
file.{o,obj}	object file
a.{out,exe}	loaded output
/tmp/lcc*	temporary files
$LCCDIR/cpp	preprocessor
$LCCDIR/rcc	compiler
$LCCDIR/liblcc.{a,lib}	*lcc*-specific library
/lib/crt0.o	runtime startup (UNIX)
/lib/[gm]crt0.o	startups for profiling (UNIX)
/lib/libc.a	standard library (UNIX)
$LCCDIR/include	ANSI standard headers
/usr/local/include	local headers
/usr/include	traditional headers
prof.out	file produced for *bprint*(1)
mon.out	file produced for *prof*(1)
gmon.out	file produced for *gprof*(1)

lcc predefines the macro **__LCC__** on all systems. It may also predefine some installation-dependent symbols; option **−v** exposes them.

SEE ALSO

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.

BUGS

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

The ANSI standard headers conform to the specifications in the Standard, which may be too restrictive for some applications, but necessary for portability. Functions given in the ANSI headers may be missing from some local C libraries (e.g., wide-character functions) or may not correspond exactly to the local versions; for example, the ANSI standard stdio.h specifies that *printf*, *fprintf*, and *sprintf* return the number of characters written to the file or array, but some existing libraries don't implement this convention.

On the MIPS and SPARC, old-style variadic functions must use varargs.h from MIPS or Sun. New-style is recommended.

With **−b**, files compiled *without* **−b** may cause *bprint* to print erroneous call graphs. For example, if **f** calls **g** calls **h** and **f** and **h** are compiled with **−b**, but **g** is not, **bprint** will report that **f** called **h**. The total number of calls is correct, however.