NAME

addr2line − convert addresses into file names and line numbers

SYNOPSIS

addr2line [**−a**|**−−addresses**]
[**−b** *bfdname*|**−−target=***bfdname*]
[**−C**|**−−demangle**[*=style*]]
[**−r**|**−−no−recurse−limit**]
[**−R**|**−−recurse−limit**]
[**−e** *filename*|**−−exe=***filename*]
[**−f**|**−−functions**] [**−s**|**−−basename**]
[**−i**|**−−inlines**]
[**−p**|**−−pretty−print**]
[**−j**|**−−section=***name*]
[**−H**|**−−help**] [**−V**|**−−version**]
[addr addr ...]

DESCRIPTION

addr2line translates addresses into file names and line numbers. Given an address in an executable or an offset in a section of a relocatable object, it uses the debugging information to figure out which file name and line number are associated with it.

The executable or relocatable object to use is specified with the **−e** option. The default is the file *a.out*. The section in the relocatable object to use is specified with the **−j** option.

addr2line has two modes of operation.

In the first, hexadecimal addresses are specified on the command line, and **addr2line** displays the file name and line number for each address.

In the second, **addr2line** reads hexadecimal addresses from standard input, and prints the file name and line number for each address on standard output. In this mode, **addr2line** may be used in a pipe to convert dynamically chosen addresses.

The format of the output is **FILENAME:LINENO**. By default each input address generates one line of output.

Two options can generate additional lines before each **FILENAME:LINENO** line (in that order).

If the **−a** option is used then a line with the input address is displayed.

If the **−f** option is used, then a line with the **FUNCTIONNAME** is displayed. This is the name of the function containing the address.

One option can generate additional lines after the **FILENAME:LINENO** line.

If the **−i** option is used and the code at the given address is present there because of inlining by the compiler then additional lines are displayed afterwards. One or two extra lines (if the **−f** option is used) are displayed for each inlined function.

Alternatively if the **−p** option is used then each input address generates a single, long, output line containing the address, the function name, the file name and the line number. If the**−i** option has also been used then any inlined functions will be displayed in the same manner, but on separate lines, and prefixed by the text **(inlined by)**.

If the file name or function name can not be determined, **addr2line** will print two question marks in their place. If the line number can not be determined,**addr2line** will print 0.

OPTIONS

The long and short forms of options, shown here as alternatives, are equivalent.

−a

--addresses
Display the address before the function name, file and line number information. The address is printed with a **0x** prefix to easily identify it.

-b *bfdname*
--target=*bfdname*
Specify that the object-code format for the object files is *bfdname*.

-C
--demangle[=*style***]**
Decode (*demangle*) low-level symbol names into user-level names. Besides removing any initial underscore prepended by the system, this makes C++ function names readable. Different compilers have different mangling styles. The optional demangling style argument can be used to choose an appropriate demangling style for your compiler.

-e *filename*
--exe=*filename*
Specify the name of the executable for which addresses should be translated. The default file is *a.out*.

-f
--functions
Display function names as well as file and line number information.

-s
--basenames
Display only the base of each file name.

-i
--inlines
If the address belongs to a function that was inlined, the source information for all enclosing scopes back to the first non-inlined function will also be printed. For example, if `main` inlines `callee1` which inlines `callee2`, and address is from `callee2`, the source information for `callee1` and `main` will also be printed.

-j
--section
Read offsets relative to the specified section instead of absolute addresses.

-p
--pretty-print
Make the output more human friendly: each location are printed on one line. If option **-i** is specified, lines for all enclosing scopes are prefixed with **(inlined by)**.

-r
-R
--recurse-limit
--no-recurse-limit
--recursion-limit
--no-recursion-limit
Enables or disables a limit on the amount of recursion performed whilst demangling strings. Since the name mangling formats allow for an infinite level of recursion it is possible to create strings whose decoding will exhaust the amount of stack space available on the host machine, triggering a memory fault. The limit tries to prevent this from happening by restricting recursion to 2048 levels of nesting.

The default is for this limit to be enabled, but disabling it may be necessary in order to demangle truly complicated names. Note however that if the recursion limit is disabled then stack exhaustion is possible and any bug reports about such an event will be rejected.

The **-r** option is a synonym for the **--no-recurse-limit** option. The **-R** option is a synonym for the **--recurse-limit** option.

Note this option is only effective if the **−C** or **−−demangle** option has been enabled.

@*file*

Read command-line options from *file*. The options read are inserted in place of the original *@file* option. If*file* does not e xist, or cannot be read, then the option will be treated literally, and not removed.

Options in *file* are separated by whitespace. A whitespace character may be included in an option by surrounding the entire option in either single or double quotes. Any character (including a backslash) may be included by prefixing the character to be included with a backslash. The *file* may itself contain additional *@file* options; any such options will be processed recursively.

SEE ALSO

Info entries for *binutils*.

COPYRIGHT