NAME

opt – LLVM optimizer

SYNOPSIS

opt [*options*] [*filename*]

DESCRIPTION

The **opt** command is the modular LLVM optimizer and analyzer. It takes LLVM source files as input, runs the specified optimizations or analyses on it, and then outputs the optimized file or the analysis results. The function of **opt** depends on whether the **–analyze** option is given.

When **–analyze** is specified, **opt** performs various analyses of the input source. It will usually print the results on standard output, but in a few cases, it will print output to standard error or generate a file with the analysis output, which is usually done when the output is meant for another program.

While **–analyze** is *not* given, **opt** attempts to produce an optimized output file. The optimizations available via **opt** depend upon what libraries were linked into it as well as any additional libraries that have been loaded with the **–load** option. Use the **–help** option to determine what optimizations you can use.

If *filename* is omitted from the command line or is **–**, **opt** reads its input from standard input. Inputs can be in either the LLVM assembly language format (.ll) or the LLVM bitcode format (.bc).

If an output filename is not specified with the **–o** option, **opt** writes its output to the standard output.

OPTIONS

–f
Enable binary output on terminals. Normally, **opt** will refuse to write raw bitcode output if the output stream is a terminal. With this option, **opt** will write raw bitcode regardless of the output device.

–help
Print a summary of command line options.

–o *filename*
Specify the output filename.

–S
Write output in LLVM intermediate language (instead of bitcode).

–{passname}
opt provides the ability to run any of LLVM's optimization or analysis passes in any order. The **–help** option lists all the passes available. The order in which the options occur on the command line are the order in which they are executed (within pass constraints).

–std–compile–opts
This is short hand for a standard list of *compile time optimization* passes. This is typically used to optimize the output from the llvm-gcc front end. It might be useful for other front end compilers as well. To discover the full set of options available, use the following command:

```
llvm-as < /dev/null | opt -std-compile-opts -disable-output -debug-pass=Argume
```

–disable–inlining
This option is only meaningful when **–std–compile–opts** is given. It simply removes the inlining pass from the standard list.

–disable–opt
This option is only meaningful when **–std–compile–opts** is given. It disables most, but not all, of the **–std–compile–opts**. The ones that remain are **–verify**, **–lower–setjmp**, and **–funcresolve**.

–strip–debug
This option causes opt to strip debug information from the module before applying other optimizations. It is essentially the same as **–strip** but it ensures that stripping of debug information is done first.

–verify–each
This option causes opt to add a verify pass after every pass otherwise specified on the command line (including **–verify**). This is useful for cases where it is suspected that a pass is creating an invalid module but it is not clear which pass is doing it. The combination of **–std–compile–opts** and **–verify–each** can quickly track down this kind of problem.

−profile−info−file *filename*
Specify the name of the file loaded by the −profile−loader option.

−stats
Print statistics.

−time−passes
Record the amount of time needed for each pass and print it to standard error.

−debug
If this is a debug build, this option will enable debug printouts from passes which use the *DEBUG()* macro. See the **LLVM Programmer's Manual**, section *#DEBUG* for more information.

−load=*plugin*
Load the dynamic object *plugin*. This object should register new optimization or analysis passes. Once loaded, the object will add new command line options to enable various optimizations or analyses. To see the new complete list of optimizations, use the **−help** and **−load** options together. For example:

```
opt -load=plugin.so -help
```

−p Print module after each transformation.

EXIT STATUS

If **opt** succeeds, it will exit with 0. Otherwise, if an error occurs, it will exit with a non-zero value.

AUTHORS

Maintained by the LLVM Team (<http://llvm.org/>).