NAME

bugpoint - automatic test case reduction tool

SYNOPSIS

bugpoint [*options*] [*input LLVM ll/bc files*] [*LLVM passes*] **--args** *program arguments*

DESCRIPTION

bugpoint narrows down the source of problems in LLVM tools and passes. It can be used to debug three types of failures: optimizer crashes, miscompilations by optimizers, or bad native code generation (including problems in the static and JIT compilers). It aims to reduce large test cases to small, useful ones. For more information on the design and inner workings of **bugpoint**, as well as advice for using bugpoint, see *LLVM bugpoint tool: design and usage* in the LLVM distribution.

OPTIONS

--additional-so *library*

Load the dynamic shared object *library* into the test program whenever it is run. This is useful if you are debugging programs which depend on non-LLVM libraries (such as the X or curses libraries) to run.

--append-exit-code=*{true,false}*

Append the test programs exit code to the output file so that a change in exit code is considered a test failure. Defaults to false.

--args *program args*

Pass all arguments specified after **--args** to the test program whenever it runs. Note that if any of the *program args* start with a "**-**", you should use:

bugpoint [bugpoint args] --args -- [program args]

The "**--**" right after the **--args** option tells **bugpoint** to consider any options starting with "**-**" to be part of the **--args** option, not as options to **bugpoint** itself.

--tool-args *tool args*

Pass all arguments specified after **--tool-args** to the LLVM tool under test (**llc**, **lli**, etc.) whenever it runs. You should use this option in the following way:

bugpoint [bugpoint args] --tool-args -- [tool args]

The "**--**" right after the **--tool-args** option tells **bugpoint** to consider any options starting with "**-**" to be part of the **--tool-args** option, not as options to **bugpoint** itself. (See **--args**,

above.)

--safe-tool-args *tool args*
Pass all arguments specified after **--safe-tool-args** to the "safe" execution tool.

--gcc-tool-args *gcc tool args*
Pass all arguments specified after **--gcc-tool-args** to the invocation of **gcc**.

--opt-args *opt args*
Pass all arguments specified after **--opt-args** to the invocation of **opt**.

--disable-{dce,simplifycfg}
Do not run the specified passes to clean up and reduce the size of the test program. By default, **bugpoint** uses these passes internally when attempting to reduce test programs. If you're trying to find a bug in one of these passes, **bugpoint** may crash.

--enable-valgrind
Use valgrind to find faults in the optimization phase. This will allow bugpoint to find otherwise asymptomatic problems caused by memory mis-management.

-find-bugs
Continually randomize the specified passes and run them on the test program until a bug is found or the user kills **bugpoint**.

-help
Print a summary of command line options.

--input *filename*
Open *filename* and redirect the standard input of the test program, whenever it runs, to come from that file.

--load *plugin*
Load the dynamic object *plugin* into **bugpoint** itself. This object should register new optimization passes. Once loaded, the object will add new command line options to enable various optimizations. To see the new complete list of optimizations, use the **-help** and **--load** options together; for example:

```
bugpoint --load myNewPass.so -help
```

--mlimit *megabytes*

Specifies an upper limit on memory usage of the optimization and codegen. Set to zero to disable the limit.

--output *filename*
Whenever the test program produces output on its standard output stream, it should match the contents of *filename* (the "reference output"). If you do not use this option, **bugpoint** will attempt to generate a reference output by compiling the program with the "safe" backend and running it.

--run-{int,jit,llc,custom}
Whenever the test program is compiled, **bugpoint** should generate code for it using the specified code generator. These options allow you to choose the interpreter, the JIT compiler, the static native code compiler, or a custom command (see **--exec-command**) respectively.

--safe-{llc,custom}
When debugging a code generator, **bugpoint** should use the specified code generator as the "safe" code generator. This is a known-good code generator used to generate the "reference output" if it has not been provided, and to compile portions of the program that as they are excluded from the testcase. These options allow you to choose the static native code compiler, or a custom command, (see **--exec-command**) respectively. The interpreter and the JIT backends cannot currently be used as the "safe" backends.

--exec-command *command*
This option defines the command to use with the **--run-custom** and **--safe-custom** options to execute the bitcode testcase. This can be useful for cross-compilation.

--compile-command *command*
This option defines the command to use with the **--compile-custom** option to compile the bitcode testcase. The command should exit with a failure exit code if the file is "interesting" and should exit with a success exit code (i.e. 0) otherwise (this is the same as if it crashed on "interesting" inputs).

This can be useful for testing compiler output without running any link or execute stages. To generate a reduced unit test, you may add CHECK directives to the testcase and pass the name of an executable compile-command script in this form:

```
#!/bin/sh
llc "$@"
not FileCheck [bugpoint input file].ll < bugpoint-test-program.s
```

This script will "fail" as long as FileCheck passes. So the result will be the minimum

bitcode that passes FileCheck.

--safe-path *path*
This option defines the path to the command to execute with the **--safe-{int,jit,llc,custom}** option.

--verbose-errors=*{true,false}*
The default behavior of bugpoint is to print "<crash>" when it finds a reduced test that crashes compilation. This flag prints the output of the crashing program to stderr. This is useful to make sure it is the same error being tracked down and not a different error that happens to crash the compiler as well. Defaults to false.

EXIT STATUS

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

SEE ALSO

opt(1)

AUTHOR

Maintained by the LLVM Team (https://llvm.org/).

COPYRIGHT