How to Build and Run an LLVM Pass

The following instructions explain how to build the template LLVM Pass in your verifier account. If you have any trouble, please post a question on the CS458 github issue page.

Build LLVM Passes

1. There are two LLVM Passes in hw4-llvm.tar.gz.

   A. IntWrite pass – an example pass that prints out all store instructions.
   To build the pass, execute the following line.

```
hw4-llvm$ make intwrite
```

   Then, `IntWrite` Pass is compiled as a shared library and stored in the LLVM library directory (i.e., `hw4-llvm/lib/libintwrite.so`).
   The runtime module (an object file) that will be used by the pass is also compiled in the same directory (i.e., `hw4-llvm/lib/intwrite-rt.o`).

   B. CCov pass – the LLVM IR pass you have to write to measure branch coverage.
   To build the pass, execute the following line.

```
hw4-llvm$ make ccov
```

   Then, it will generate similar files in `hw4-llvm/lib` as like `IntWrite`.

Run LLVM Passes

1. You can instrument a target program with an LLVM Pass as you compile the target program. You can configure clang to run a given LLVM Pass in the middle of the compiling process, such that the produced binary gets modified. If the LLVM Pass inserts a new function declaration with its definition, you should link the object file with the definition.

For example, you can build `test/example.c` with `IntWrite` instrumentation as follows:

```
hw4-llvm$ cd test
hw4-llvm/test$ clang --ld-path=ld.lld -fno-experimental-new-pass-manager
      example.c ../lib/intwrite-rt.o -g -O0 -Xclang -load -
      Xclang ../lib/libintwrite.so -o ./example
```

Note that `-g` option was used for `IntWrite` to utilize the debugging information (e.g. line numbers).

As you execute `./example`, you can see that log file is produced by the probe executions.

```
hw4-llvm/test$ ./example
hw4-llvm/test$ cat log
Store value 0 in unknown location
Store value 0 in Function main, line 15
Store value 0 in unknown location
Store value 2 in Function f1, line 5
Store value 4 in Function f1, line 10
Store value 4 in Function main, line 16
Store value 4 in Function main, line 24
Store value 0 in Function main, line 28
Store value 4 in Function main, line 29
Store value 1 in Function main, line 28
Store value 5 in Function main, line 29
Store value 2 in Function main, line 28
Store value 7 in Function main, line 29
Store value 3 in Function main, line 28
Store value 10 in Function main, line 29
Store value 4 in Function main, line 28
Store value 14 in Function main, line 29
Store value 5 in Function main, line 28
Store value 19 in Function main, line 29
Store value 6 in Function main, line 28
Store value 25 in Function main, line 29
Store value 7 in Function main, line 28
Store value 32 in Function main, line 29
Store value 8 in Function main, line 28
Store value 40 in Function main, line 29
Store value 9 in Function main, line 28
Store value 49 in Function main, line 29
Store value 10 in Function main, line 28
Store value 98 in Function main, line 34
Store value 196 in Function main, line 34
Store value 197 in Function main, line 48
```

Similarly, you can run `IntWrite` for `grep.c` as follows:

```
hw4-llvm$ cd test/grep
hw4-llvm/test/grep$ clang --ld-path=ld.lld -fno-experimental-new-pass-
      manager grep.c ../../lib/intwrite-rt.o -g -O0 -Xclang -load -
      Xclang ../../lib/libintwrite.so -o grep
```

The above commands are written in the makefile for your convenience.

```
hw4-llvm$ make intwrite_test
```

You can run your `CCov` pass as the similar way.