1 logs

```
sbmake -distcc DEBUG=1 CTB="sl_main sl_compile
shared_simulink_block sl_mask sl_libraries sl_link_bd
simulink_core"
\begin{itemize}
    \item type punning using union, reinterpret case, strict alialing rule
    \item union type punning, size of constexpr int == 4B or less than 4B afl-fuzz
    \item hardening
    \item ADL
    \item exceptions:https://monoinfinito.wordpress.com/series/exception-handling-in-c/
        https://itanium-cxx-abi.github.io/cxx-abi/
        http://www.logix.cz/michal/devel/gas-cfi/dwarf-2.0.0.pdf
https://lse.epita.fr/data/lt/2016-12-13/lt-2016-12-13-Francis_Visoiu_Mistrih-c++_internals.p
https://gcc.gnu.org/onlinedocs/libstdc++/manual/using_exceptions.html
https://github.com/gcc-mirror/gcc/blob/master/libstdc%2B%2B-v3/include/std/any


https://blog.regehr.org/archives/1520
\end{itemize}

std::bitset<5> intA;
struct s1 {
    int a:6;
}
#include <iostream>
#include <stdexcept>
#include <functional>
#include <bitset>
#include <locale>
#include <stdio.h>
#include <codecvt>
#include <assert.h>
using namespace std::string_literals;
auto s1 = "abc";
auto s1 = L"abc";
auto s1 = u8"abc";
auto s1 = u"abc";
auto s1 = U"abc";
auto s1 = "abc";
auto s1 = "abc";
auto s1 = 42;
auto s1 = 42l;
auto s1 = 42ul;
auto s1 = 42ull;
```

```
auto s1 = 42.f;
auto s1 = 42.;
auto s1 = 42;
auto s1 = 0b11
auto s1 = 0123;
auto s1 = 0x123;
std::cout << std::hex << std::dec << std::oct << std::boolalpha
std::cout << std::bitset<5>(19) << std::endl;
std::setlocale(LC_ALL, "en_US.UTF-8");
std::u32string p1 = U"abc"s;
std::string p2 = "abc"s;
std::u16string p3 = u"abc"s;
enum e1{R,RED,YELLOW};
enum class e1{R,RED,YELLOW};
#define RED1 1
enum class e1:uint8_t
{
    R=253,
    RED,
    YELLOW
};

template <typename T> void type_name(){
    std::cout << __PRETTY_FUNCTION__ << std::endl;
}
string toUTF8(const basic_string<T, char_traits<T>, allocator<T>>& source)
{
    string result;
    wstring_convert<codecvt_utf8_utf16<T>, T> convertor;
    result = convertor.to_bytes(source);
    return result;
};

template <typename T>
auto fromUTF8(string source)
{
    wstring_convert<codecvt_utf8_utf16<T>, T> convertor;
    basic_string<T, char_traits<T>, allocator<T>>  result =
    convertor.from_bytes(source); return result; }

//UB, no exception
 int x = 20;
 auto p = reinterpret_cast<float*>(&x);

    cout <<  (void*)main << endl;
```

```
union
   {
       float input; // assumes sizeof(float) == sizeof(int)
       int   output;
   } data;
   data.input =1*sqrt(-1);//1*pow(2,-127-22)/2;///1e-45;//0b00000001000000000000001000000000
   cout << data.input << endl;

   std::bitset<sizeof(float) * CHAR_BIT> bits(data.output);
   std::cout <<  bits << std::endl;

   // or
   std::cout << "BIT 4: " << bits[4] << std::endl;

   print floating point raw bits
   auto floatValue = numeric_limits<float>::max();
   int* intVal = reinterpret_cast<int*>(&floatValue);
   std::bitset<sizeof(float) * 8> bits1(*intVal);
```

```
1. shared folder ubuntu guest:
sudo vmhgfs-fuse -o nonempty -o allow_other .host:/data ./testmnt
2. cmake llvm
cmake -DCMAKE_EXPORT_COMPILE_COMMANDS=ON -DCMAKE_VERBOSE_MAKEFILE=ON ../llvm
-DLLVM_ENABLE_PROJECTS='clang;clang-tools-extra;lldb;compiler-rt;lld;polly'
```

```
 >> !printenv LD_LIBRARY_PATH
```

```
#include <stdio.h>
#include <stddef.h>
#include "rtwtypes.h"
#include "slx1.h"
extern void slx1_initialize(void);
extern void slx1_step(void);
extern void slx1_terminate(void);
```

```
//create slx model named "slx1" with outport named "Outport" and generated code and package
//gcc -L./slx1/debug2/ -I./slx1/matlab/simulink/include -I./slx1/matlab/rtw/c/src -I./slx1/c
//LD_LIBRARY_PATH=/home/ppatil/Downloads/debug2/slx1/debug2:$LD_LIBRARY_PATH
int main()
{
    int i;
    scanf("%d", &i);
    printf("abdczs start\n");
    slx1_initialize();
    for(int v=0;v<i;++v){
        slx1_step();
        printf("v12 %f\n", slx1_Y.Outport);
    }

    slx1_terminate();
}

/* End of file */
```