

# 1 logs

```
\begin{itemize}
  \item type punning using union, reinterpret case, strict aliasing 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.pdf
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;

```



```

{
    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 */

```