C++ Toolchain

Justin M. Turney

Linux Distros

Distro	Released	Compiler	EOL
RHEL 7	June 2014	gcc 4.8	June 2024
RHEL 7.4	August 2017	gcc 4.8	June 2024
SLES 11	March 2009	gcc 4.3	March 2022
SLES 12	October 2014	gcc 4.8	October 2027
Ubuntu 14.04 LTS	April 2015	gcc 4.8/clang 3.4	April 2019
Ubuntu 16.04 LTS	April 2016	gcc 5/clang 3.8	April 2021

GCC Releases

Release	Release Date
4.8.0 / 4.8.5	March 2013 / June 2015
5.3	December 2015
6.4*	July 2017
7.2*	August 2017

*** Officially supported versions**

LLVM Releases

Release	Release Date
3.4	January 2014
3.8	March 2016
5.0	September 2017

Release Cycles

- 6 months to 1 year for a C++ toolchain release.
- 6 months to N-years for a distro release
- Unknown amount of time for someone to then install distro on systems.

What’s the alternative?

- Compile the toolchain on your own…

What’s the alternative?

- LLVM/Clang is very easy to compile.
  - https://clang.llvm.org/get_started.html

Address Sanitizer (asan)

Asan is a memory error detector for C/C++.

1. Use after free (dangling pointer dereference*
2. Heap buffer overflow*
3. Stack buffer overflow
4. Global buffer overflow
5. Use after return
6. Use after scope
7. Initialization order bugs
8. Memory leaks* (on Linux)

*** Only errors Valgrind/Memcheck can detect.**

Examples

Example 1

```
// To compile: clang++ -g -fsanitize=address asan1.cpp -o asan1
int main(int argc, char **argv)
{
    int *array = new int[100];
    delete[] array;
    return array[argc]; // BOOM
}
```

Example 2

```
// To compile: clang++ -g -fsanitize=address asan2.cpp -o asan2
int main(int argc, char **argv)
{
    int *array = new int[100];
    array[0] = 0;
    int res = array[argc + 100]; //BOOM
    delete[] array;
    return res;
}
```

Example 3

```
// To compile: clang++ -g -fsanitize=address asan3.cpp -o asan3
int main(int argc, char **argv)
{
    int stack_array[100];
    stack_array[1] = 0;
    return stack_array[argc + 100]; //BOOM
}
```

Example 4

```
// To compile: clang++ -g -fsanitize=address asan4.cpp -o asan4
int global_array[100] = {-1};

int main(int argc, char **argv)
{
    return global_array[argc + 100]; // BOOM
}
```

Example 5

```
// To compile: clang++ -fsanitize=leak -g asan5.cpp -o asan5
#include <stdlib.h>

void *p;

int main() {
  p = malloc(7);
  p = 0; // The memory is leaked here.
  return 0;
}
```

YouTube

- Learn C++ through conference videos on YouTube.
  - C++Con
  - PacifiC++
  - LLVM