CPPCON 2018 NOTES

by Anny G.
@annygakn
twitter

Bjarne Stroustrup - monday, sept 24, 2018, cppcon 2018

CONCEPTS: THE FUTURE OF GENERIC PROGRAMMING (the future is here)

This is not a talk oriented at technical language details.

Individual features in isolation from language are not the intent here

The goal is to improve as simple as generic programming code, but for that becomes we need to improve non generic other features so it

Concepts

concepts vs types

Type

- specifies how you can use an object
- not just for functions, classes

- specifies set of operations for an object

concepts gives us this

⊕ readability, maintainability

conditional properties

"make it so → operator can only be used if other is a class"

Q: Can't we use std::enable_if ?
A: they don't scale! need lots of workarounds!

reliability

- auto is misused sometimes ...

e.g.

```
auto x = foo(1); // x is InputChannel
```

not maintainable :(

```
InputChannel x = foo(1);
```

⊕ readable

Re: readability

"more readable code comes from those who understand concepts, and less from those who really know the old ways and use concepts in the same way..."

Q: are concepts types of types?
A: No! most take more than one arg..

History

CONCEPTS ISN'T REALLY A NEW THING

Think of **compile time predicates**:

- they are quite fundamental
- we always had concepts

concepts before

/* documentation and other comments... */

↑ in developer's head

↑ in documentation

↑ in comments

BUT NOW WE GET TO REPRESENT THEM IN CODE

⊕ concepts are flexible and allow you to keep extending and working **using C++** on them because you probably aren't get it right the first time

⊕ once you use concepts you don't go back - you start thinking in terms of them ⊕ you express ideas more clearly so even A COMPILER CAN UNDERSTAND!

this change is not a detail but a change in how we THINK! Major change → might take a while :(

Matthew Butler

monday, sept 24, 2018, cppcon

Secure Coding Best Practices

3 LIES WE TELL EACH OTHER

~~firewalls~~ - didn't save other companies ...

~~code review, tests~~ - code reviews are not focused on finding exploits

~~we are too~~ - big ⇒ lots to lose {small, large, etc} to be a target

small ⇒ too weak

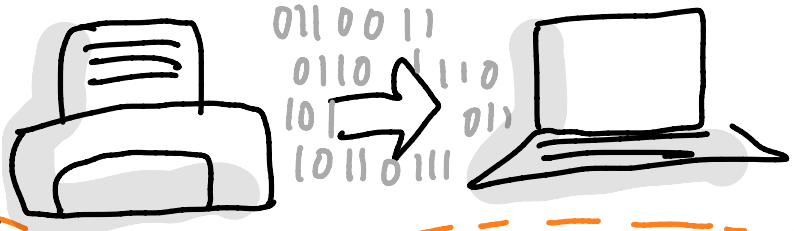
— you are always a target

CRITICAL SYSTEMS

... what is it?

... any other systems that can interact with it no matter how low the priority

e.g. PRINTER



Anything one from outside can get into →

ATTACK VECTORS!

e.g. command line interfaces

Need to (1) validate incoming arguments (2) study the standard

and remember

warnings are errors! Listen to them

WHAT I LOOK FOR WHEN PENETRATING A SYSTEM

- ☐ copy/moves of memory
- ☐ not validating your data or verifying the sender
- ☐ open source libraries
  their weaknesses are your weaknesses
- ☐ unguarded internal interfaces
  don't assume inside systems to be safe
- ☐ complexity in your design

some STRATEGIES

ASLR - randomize address space

STACK CANARIES
- put known bit pattern (check before returning)
- abort if incorrect
- created by compiler

To read up
- ☐ ASLR
- ☐ return oriented programming

Simon Brand - monday, sept 24, 2018, cppcon2018

I have been really excited for this talk!!! I care a lot about debuggers and I find them a neat piece of SW.

How C++ Debuggers Work

ELF
binary format for executables! also has headers

DWARF
- debug info format
- describes stuff about your program, e.g. functions, types, etc

Sometimes ELF is enough for your debugger but usually for complicated things it needs more info

debuggers will consume DWARF

HARDWARE BREAKPOINTS
- special registers - limited
- can break on read/write/execute

SOFTWARE BREAKPOINTS
- modify running code so breakpoint occurs
- unlimited - only break on execute

x86 HW bpoint
- 4 registers and you write breakpoint address

x86 SW bpoint
- replaces the instruction with int 3

I forget what Simon said about what happens to the replaced instruction... TODO look at exact details

QUESTIONS
1. how is dwarf generated
2. what exactly happens when an instruction is replaced with 'int 3'

MANGLING

- each function gets mangled differently
- differs from compiler to compiler

QUESTION

- how does breakpointing on specific lines?

STEPPING

STEP OUT: break point on return

STEP IN: return addr, add of next instr

?? I wrote this but forgot what it means

process_vm_ready

reads more than one page @ a time

MISC

DWARF gives you info about inlined functions

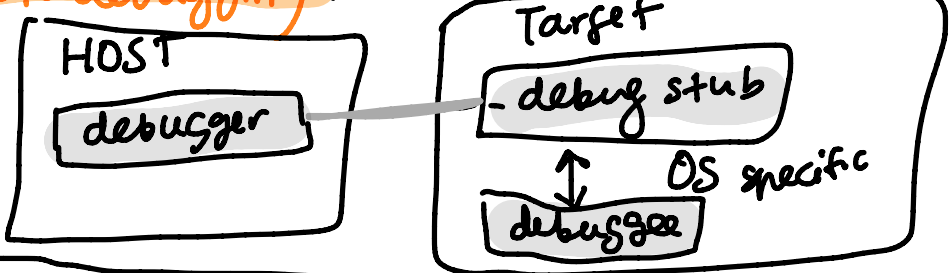
DW_TAG_subprogram

- each func in DWARF has entries for beginning and end of it so you can compare PC ranges for each function and figure out the exact name of current function

Shared libraries

- r_debug
  link_map ← tells you of all shared libraries
  r_brk_ ← gets called when a library is being called

remote debugging



a lot of debuggers work like this

Kate Gregory

monday, sept 24, cppcon 2018

WHAT DO WE MEAN WHEN WE SAY NOTHING AT ALL

WRITE YOUR CODE AS IF YOUR CHILDREN WILL BE MAINTAINING IT.

Use your own judgement

[[fallthrough]]

it doesn't make a difference at runtime but the compiler will complain if you misuse it

• BUT USE YOUR OWN JUDGEMENT.

on defaults

- avoid defaults because not everyone knows it and it communicates your intent better
- saves others time guessing if you have considered it

• FOR NOTHING TO MEAN SOMETHING, IN OTHER PLACES YOU HAVE TO MAKE YOUR STATEMENTS

EXPLICIT

parameter passing

const, & convey the meaning about your API design

e.g. Orders, customers

traditional loop

- when you have a traditional loop, is it because it's not touching some objects ??
  → there are algorithms for that ... there are lambdas too..

initialization

what do you mean when you initialize something to its default value?

clear code communicates w/ future readers

shows them why you did this

- use your own judgement -

COMMUNICATE

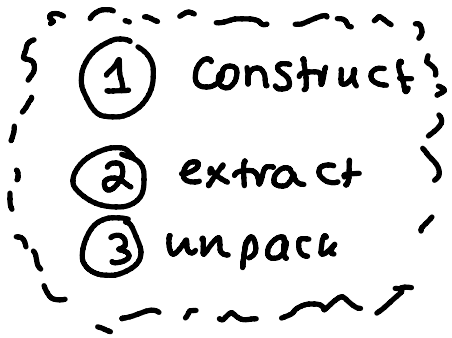
"what does it mean when you make such choices"

ENSURE YOUR NOTHING speaks volumes

Richard Powell — tuesday, sept 25, cppcon 2018

NAMED ARGUMENTS IN C++ FROM SCRATCH

what if we had a map with args and their names?..



← Rough Idea

Random Question from Anny ...
Q: what does header only library mean
hmmm...
why is it important

"Find a question: can I do this in/with C++?"

! not everything can be production ready

! the most important thing is the JOURNEY AND LEARNING THINGS

to look up
- hana boost
- hana literals

Patricia Aas

wednesday, sept 26, cppcon 2018

SOFTWARE VULNERABILITIES

dont depend on the undefined behaviour

sometimes compilers optimize away the MEMSET (important if you're depending on it to erase your passwd

shellcode - piece of code - today it meant to represent that you are able to execute something natively

- dont take external strings as format strings - they might not supply enough args and prntf will read whatever the next thing on stack is

to look up

- ☆ Smashing The Stack for Fun and Profit

Kate Gregory

wednesday, sept 26, cppcon 2018

SIMPLICITY: NOT JUST FOR BEGINNERS

- when you teach, start simple
  - omit error checking → lessen cognitive burden!
- encapsulation ≠ obscuration
- simpler code is harder to write:
  - → not often faster!
  - because to make it faster you have to know something about the language

e.g.

auto p vs auto p &

- short function names!
  - if you can't name it, it is probably several functions
  - (but don't go refactoring super old legacy functions that work perfectly fine...)

```
x(5,6,7,8)
```

use your judgement

```
r= a(5,6)
b(r,7,8)
```

- no NESTING → return early

- clean up in destructors
  - (+) no one will forget to clean up esp if they might add a new throw somewhere...

- # THINK ABOUT THE FUTURE

  - it might be easy now but later it will confuse people

to do

overview std algorithms

wednesday, sept 26, cppcon2018

Bryce Lelbach, Jens Weller, Jon Kalb, Phil Nash, Stephen T. Lavavej

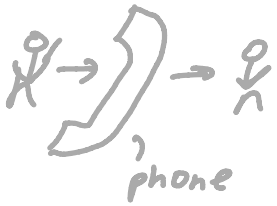
C++ Community Building

— birds of a feather session —

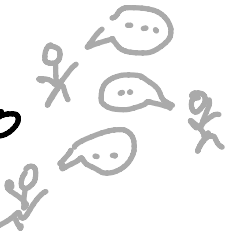
Goal: Discuss how to build a C++ community

Ideas for meetups

- reach out to speakers in other cities, in case they are travelling. If they are visiting your city, politely ask if they would like to speak at your event

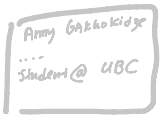


- It's important not only to have talks but also discussions and have the meetup be a group, not just a service

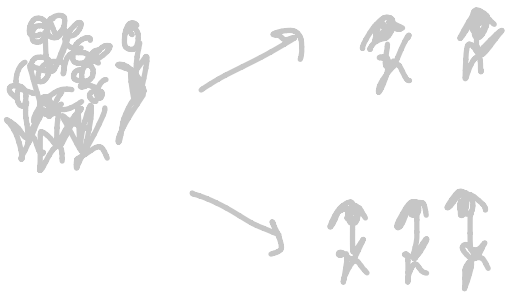


- have lighting talks
- book club?

- get people to showcase
- can review resumes?



- sometimes have activities that allow you to break up into smaller groups



- diversity of topics!

agenda
- debuggers - refactorys...
- graphics

$ponsorships

- tell the company what they will get
- tell them what you need
  e.g. pizza + flyers printed
- can provide them with 2 mins at the beginning of each talk, etc

probably need to look to work with companies who hire C++ devs

how do you get volunteers?

- should split up the roles between people
  - can rotate the roles
- assign different people to find speakers
  → lead discussions
- ask for specific things to get done
  → e.g. we need someone to order pizza, receive the delivery and setup the pizzas

! you have to let go and not micro-manage people

! appreciate volunteers

Herb Sutter Thursday, sept 27, cppcon 2018

THOUGHTS ON A MORE POWERFUL AND SIMPLER C++

When we add new things to the language, it's to

1. make user code simpler
2. remove edge cases e.g. lifetimes
3. discourage certain things - metaclasses to discourage xxx forgot

WHAT IS C++

- ☆ don't pay for what you don't use (∅ overhead)
- ☆ still pay for it if you use it.
- ☆ leave no room for lower lang (except asm)
- ☆ backw. compatibility

**

GOOD C++ FEATURE

remove gotchas

express intent better

simplify C++ code (read, write, debug)

remove an exception to **

help us deprecate something e.g. typedef → using

TO DO
watch last years previous talk

Jeff Trull

thursday, sept 27, 2018 cppcon

GDB PYTHON API

IMPROVE STACK TRACE

- Frame Decorators = Filters
- override ↑ to add custom behaviour

e.g. decorator, replace verbose names with well known values

EASILY STEP TO USER CODE

- can make it so we ignore library code
- gdb.Breakpoint
  - put commands that execute when a bpoint is encountered
  - use libclang to find out semantic info about your code

FINDING LOOPS

- use valgrind, parse output of monitor
- use find processing to find loops

ideas

- can try to take notes on things I constantly need to fix in debugger
- figure out the solution, fix it, tell others