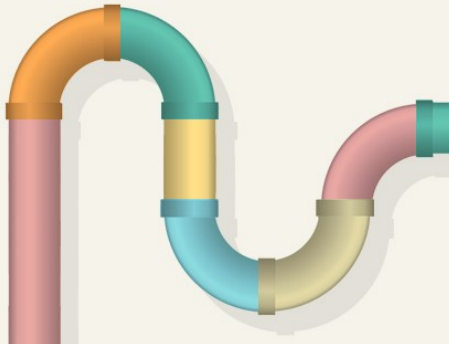
Teaching GraalVM Native DWARFish

or is it Dwarvish? whatever!

Andrew Dinn

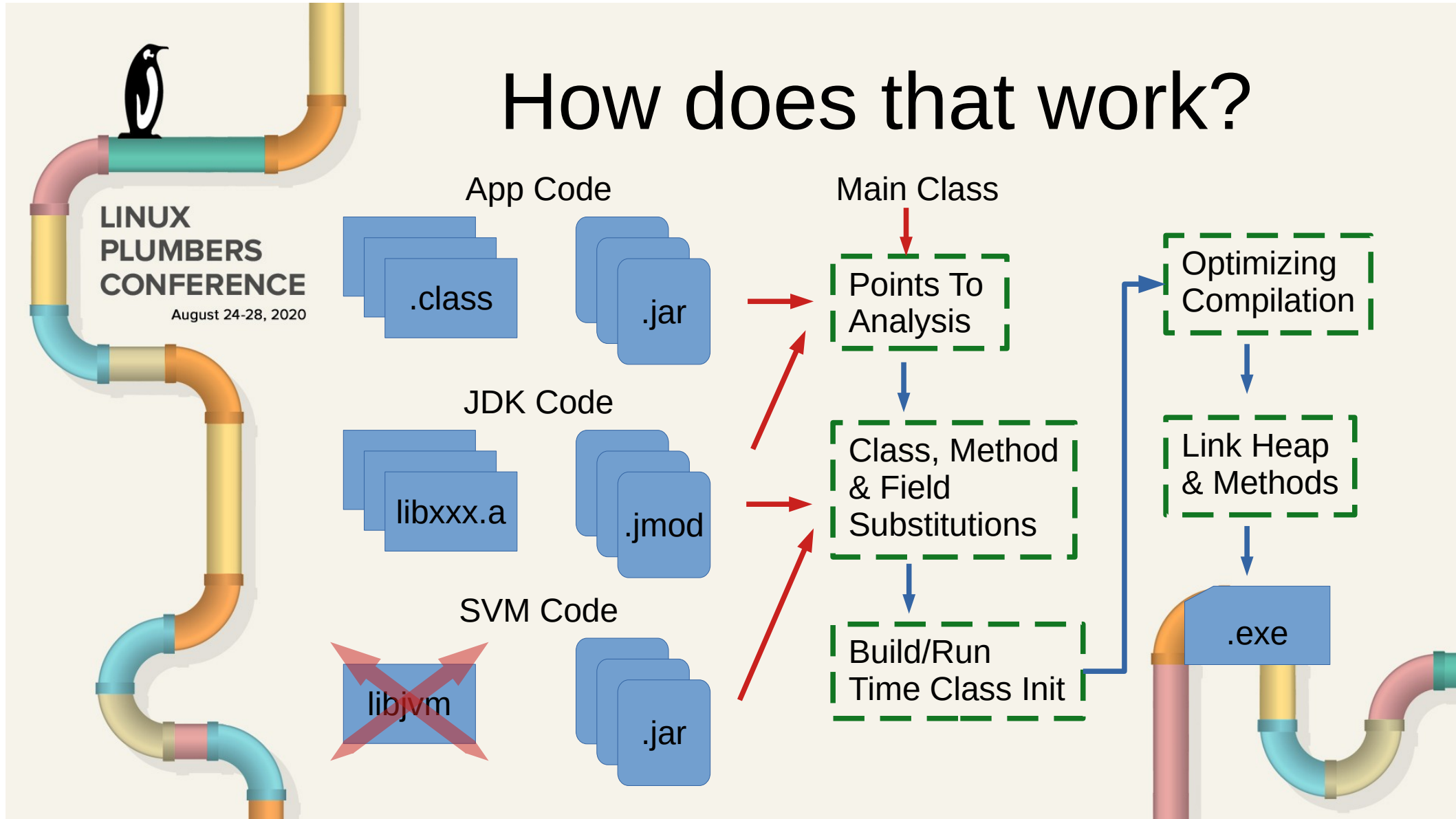
Distinguished Engineer
Red Hat Java Team



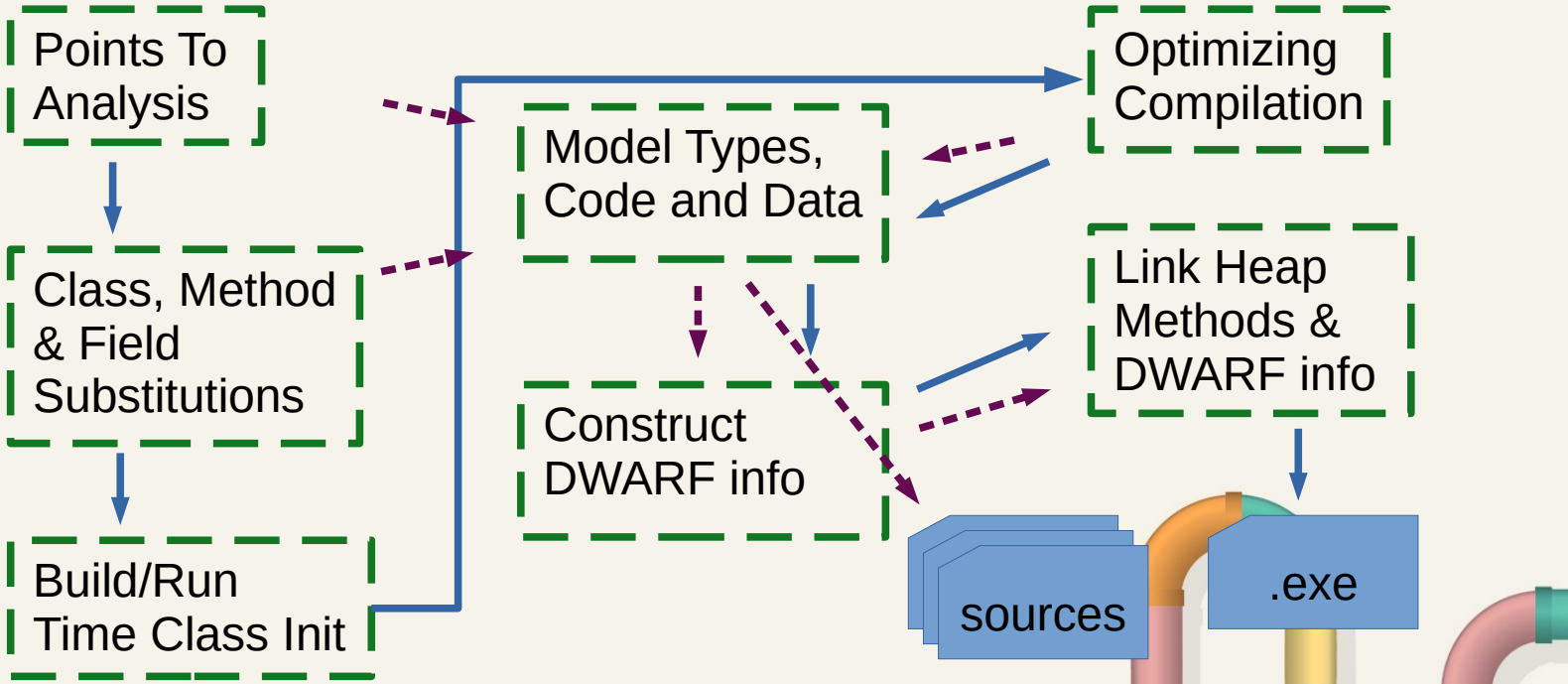
LINUX PLUMBERS CONFERENCE
August 24-28, 2020

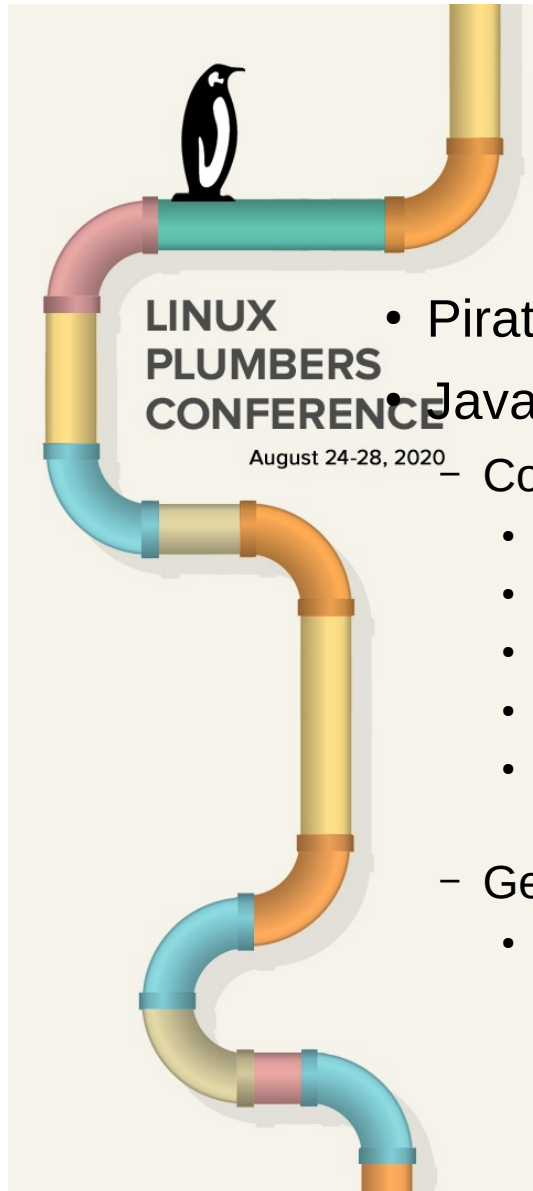
GraalVM Native

- Alternative delivery option for Java apps
  - Static (offline) compilation to self-contained binary
    - No JVM needed, but retains JDK runtime
  - No runtime class loading
    - Class loaders throw ClassNotFoundException
  - Closed world model
    - So all required classes must be presented in advance
- Is it Java?
  - Already part of an Oracle released product (GraalVM) so… yes
    - (also Red Hat mandrel releases of just GraalVM Native)
  - Semantics will be ‘regularized’ (by OpenJDK project)


LINUX
PLUMBERS
CONFERENCE
August 24-28, 2020
How does that work?
App Code
.class
.jar
JDK Code
libxxx.a
.jmod
SVM Code
libjvm
.jar
Main Class
Points To
Analysis
Class, Method
& Field
Substitutions
Build/Run
Time Class Init
Optimizing
Compilation
Link Heap
& Methods
.exe

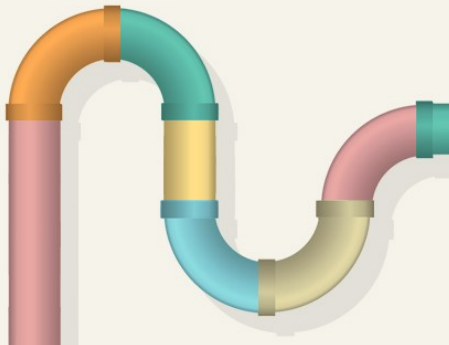
How Do We Debug It?

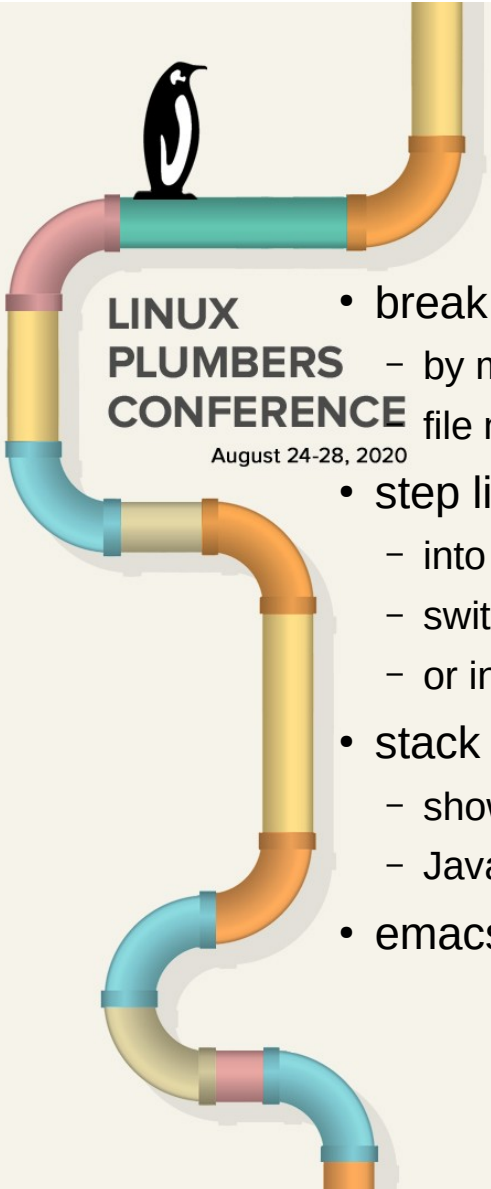




Current DWARF Model

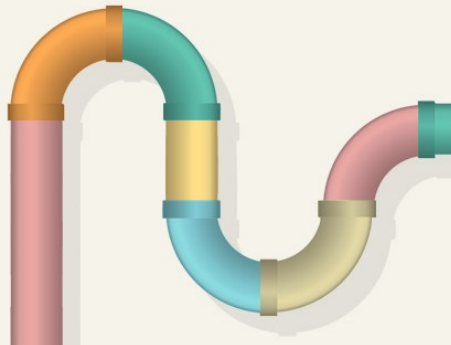
- Pirate C++ model (so gdb can understand it)
- Java → C++ Method mapping
  - Compile Unit groups methods by class
    - SUBPROGRAM per method
    - owner class, name+sig, range, visibility
    - source file + address → line map
    - frame size + extend/teardown offsets
    - inlined ranges
      - detail owner, name+sig, source file + address → line map
  - Generated DWARF Sections
    - info+abbrev, aranges, frame, line, string





Current gdb support

- break points
  - by method name or file+line
  - file names resolved via sources cache
- step line by line
  - into or over calls
  - switches to current method line
  - or inline/substituted method line
- stack backtraces
  - shows outer compiled methods when in inlined code
  - Java ↔ native transitions ‘just works’
- emacs ‘just works’



LINUX PLUMBERS CONFERENCE
August 24-28, 2020

Planned DWARF Model

- Types
  - Requires Java → C++ Type mapping
    - class Foo → typedef class _Foo *Foo
      - where class _Foo { struct ObjHeader _h; jint f1; Bar f2; … }
    - FooBar extends Foo → class _FooBar: public _Foo
    - interface → union { class _Foo *, …}
    - array of X → struct _ArrX { struct ArrHeader _h; X _elem[0] }
- Heap data
  - static fields + constants

LINUX PLUMBERS CONFERENCE
August 24-28, 2020

Planned gdb support

- print object header + contents field by field
- name, type, location (and liveness) for
  - parameter vars
  - local vars
  - static fields
- casts
- traverse object network using path exprs

Thank You

upstream GraalVM:
github.com/oracle/graal

Red Hat GraalVM Native only

https://github.com/graalvm/mandrel

my work in progress:
https://github.com/adinn/graal/tree/debugtypes

demo:
https://youtu.be/JqV-NFWupLA