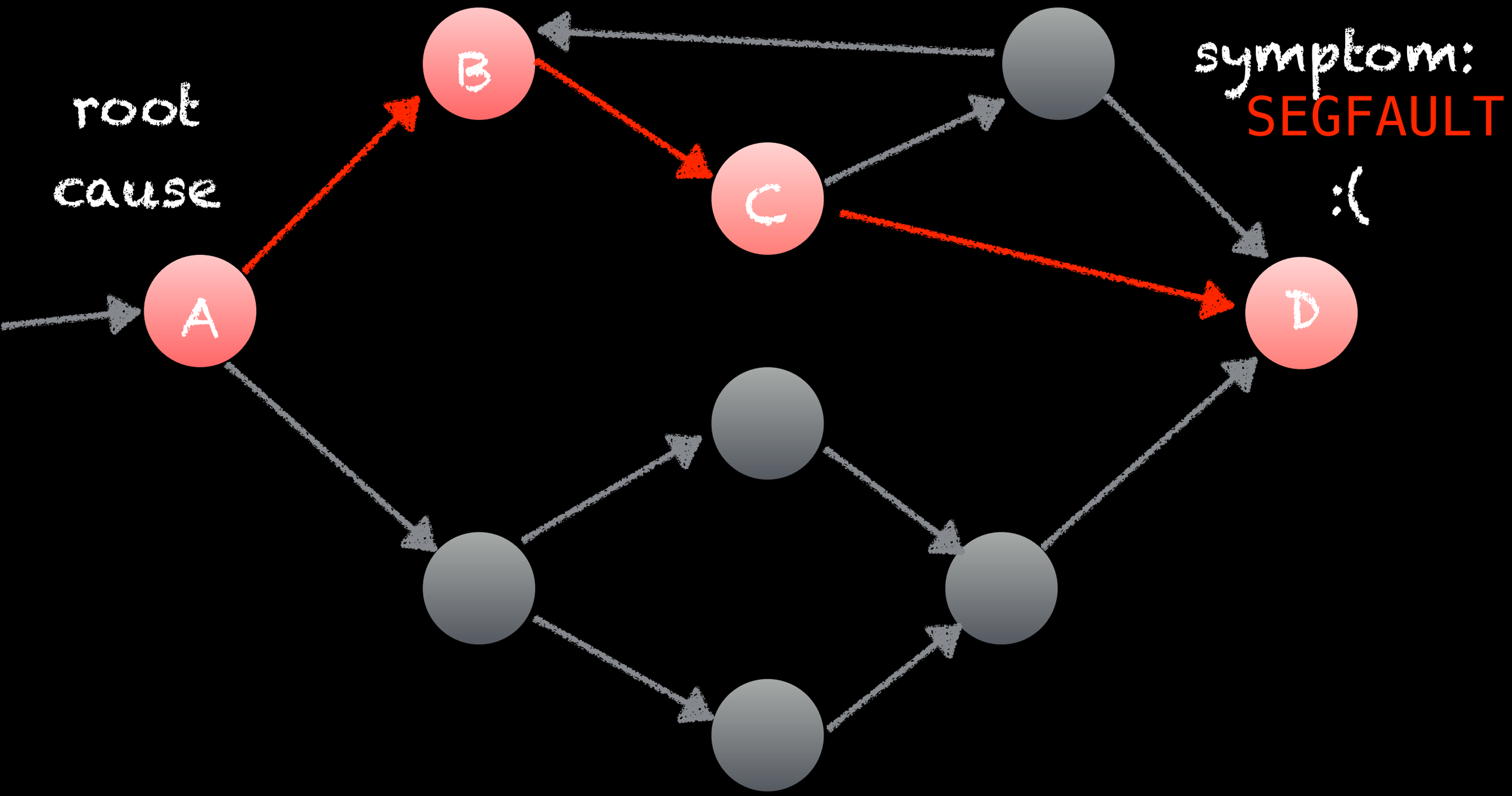
Back to the Futurre

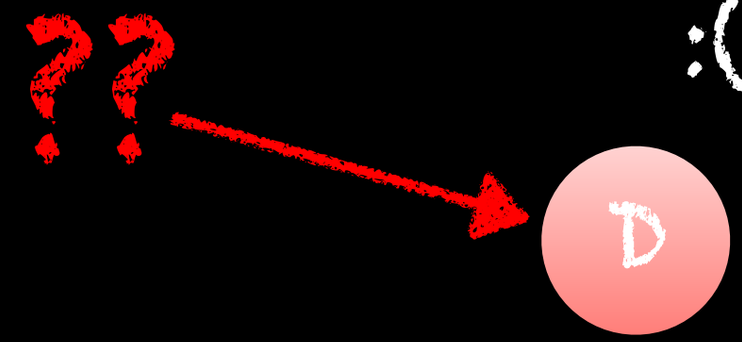
Deterministic Debugging with rr

Anatomy of a Bug



What the programmer knows

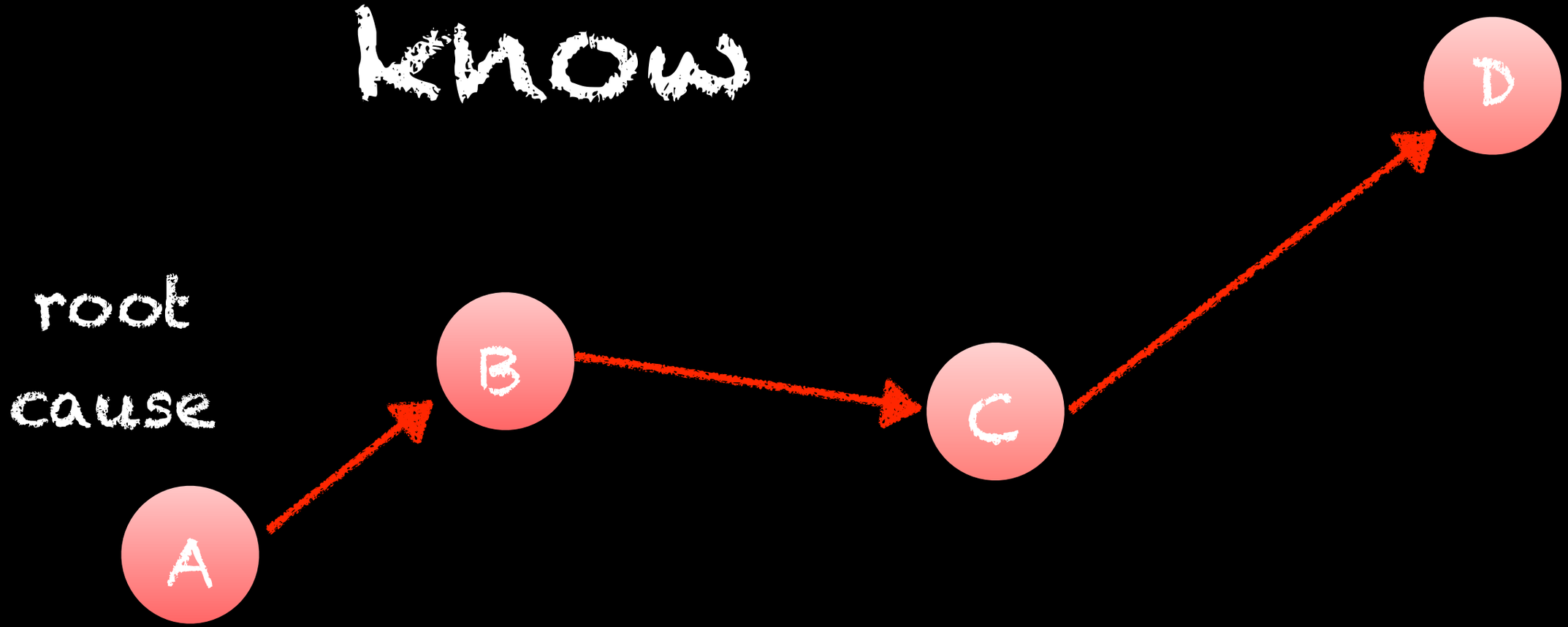
symptom:
SEGFAULT
:(



What the programmer wants to know

symptom:
SEGFAULT
:(

root cause

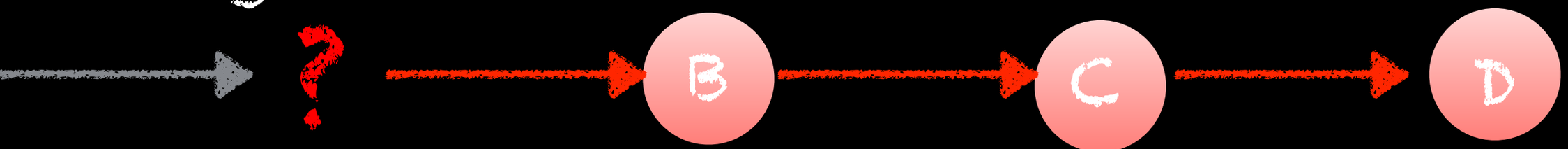


Traditional Debuggers

run

re-run

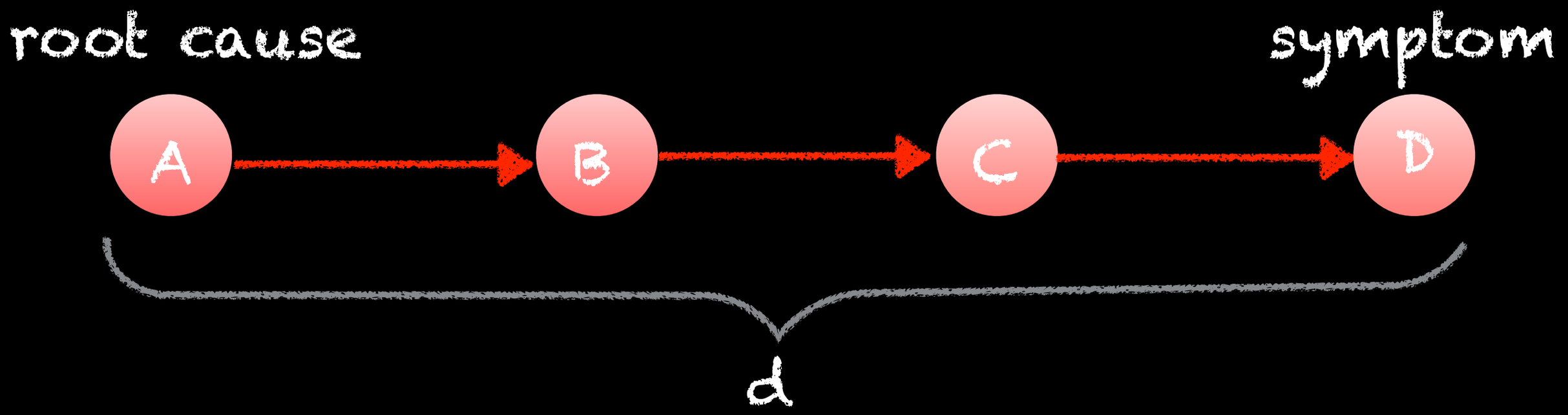
run again...



. . .

O(d*r)

d = distance between cause and symptom
r = inverse reproducibility, ie repro's 1/r executions



Enter rr

https://flic.kr/p/mYvfNr

https://flic.kr/p/7zrdKz

https://flic.kr/p/7JH3T3

$O(r+d)$

https://treeherder.mozilla.org/#/jobs?repo=fx-team

Treeherder | Sheriffing | Infra | Repos | Filters | (+) | Filter platforms & jobs

fx-team | 88 unclassified | 1

Sun Sep 27, 8:55:05 - jsantell@mozilla.com — *- Complete -*

Linux opt	? ? B Cpp Jit1 Jit2 Mn Mn-e10s Wr X M(1 2 3 4 5 JP bc1 bc2 bc3 bc4 bc5 bc6 bc7 dt1 dt2 gl oth p) M-e10s(1 2 3 4 5 bc1 bc2 bc3 bc4 bc5 bc6 bc7 dt1 dt2) R(C J R1 R2 Ru1 Ru2) R-e10s(C R-e10s) T(c d g1 g2 o s tp) T-e10s(c d g1 g2 o s tp) W(1 2 3 4)
Linux pgo	? ? B Cpp Jit1 Jit2 Mn Mn-e10s Wr X M(1 2 3 4 5 JP bc1 bc2 bc3 bc4 bc5 bc6 bc7 dt1 dt2 gl oth p) M-e10s(1 2 3 4 5 bc1 bc2 bc3 bc4 bc5 bc6 bc7 dt1 dt2) R(C J R1 R2 Ru1 Ru2) R-e10s(C R-e10s) T(c d g1 g2 o s tp) T-e10s(c d g1 g2 o s tp) W(1 2 3 4)
Linux debug	B Cpp Jit1 Jit2 Mn Wr X1 X2 M(1 2 3 4 5 bc1 bc2 bc3 bc4 bc5 bc6 bc7 dt1 dt2 dt3 dt4 dt5 dt6 dt7 dt8 gl oth p) M-e10s(1 2 3 4 5 bc1 bc2 bc3 bc4 bc5 bc6 bc7) R(C J R1 R2 R3 R4) R-e10s(R-e10s1 R-e10s2) W(1 2 3 4 5 6 7 8)
Linux x64 opt	? ? B Cpp H Jit1 Jit2 Ld Mn V Wr X M(1 2 3 4 5 JP bc1 bc2 bc3 bc4 bc5 bc6 bc7 dt1 dt2 gl oth p) M-e10s(1 2 3 4 5 bc1 bc2 bc3 bc4 bc5 bc6 bc7 dt1 dt2) R(C J R1 R2 Ru1 Ru2) R-e10s(C R-e10s) T(c d g1 g2 o s tp) T-e10s(c d g1 g2 o s tp) W(1 2 3 4) tc[Tier-2](B)
Linux x64 pgo	? ? B Cpp Jit1 Jit2 Ld Mn Wr X M(+18) M-e10s(1 2 3 4 5 bc1 bc2 bc3 bc4 bc5 bc6 bc7 dt1 dt2) R(C J R1 R2 Ru1 Ru2) R-e10s(C R-e10s) T(c d g1 g2 o s tp) T-e10s(c d g1 g2 o s tp) W(1 2 3 4)
Linux x64 asan	Bd Bo Cpp Jit1 Jit2 M(1 2 3 4 5 JP bc1 bc2 bc3 bc4 bc5 bc6 bc7 dt1 dt2 dt3 dt4 gl oth p) M-e10s(1 2 3 4 5 bc1 bc2 bc3 bc4 bc5 bc6 bc7) R(C J R1 R2)
Linux x64 debug	B Cpp Jit1 Jit2 Mn S Wr X1 X2 M(1 2 3 4 5 bc1 bc2 bc3 bc4 bc5 bc6 bc7 dt1 dt2 dt3 dt4 gl oth p) M-e10s(1 2 3 4 5 bc1 bc2 bc3 bc4 bc5 bc6 bc7) R(C J R1 R2 R3 R4) R-e10s(R-e10s1 R-e10s2) W(1 2 3 4 5 6 7 8) tc[Tier-2](B)
OS X 10.6 opt	Cpp Jit X M(1 2 3 4 5 JP bc1 bc2 bc3 bc4 bc5 bc6 bc7 dt1 dt2 gl oth p) M-e10s(bc1 bc2 bc3 bc4 bc5 bc6 bc7) R(C J R)
OS X 10.6 debug	Cpp Jit Mn X M(1 2 3 4 5 JP bc1 bc2 bc3 bc4 bc5 bc6 bc7 dt1 dt2 dt3 dt4 gl oth p) R(C J R)
OS X 10.7 opt	B
OS X 10.7 debug	B S tc[Tier-2](B)
OS X 10.10 opt	Cpp Wr X M(1 2 3 4 5 JP bc1 bc2 bc3 bc4 bc5 bc6 bc7 dt1 dt2 gl oth p) M-e10s(bc1 bc2 bc3 bc4 bc5 bc6 bc7) R(C J R) T(+7) T-e10s(c g2 o s tp) W(1 2 3 4)
OS X 10.10 debug	Cpp Mn Wr X M(1 2 3 4 5 JP bc1 bc2 bc3 bc4 bc5 bc6 bc7 dt1 dt2 dt3 dt4 gl oth p) R(C J R) W(1 2 3 4 5 6 7 8)
Windows XP opt	B Cpp Jit Mn Wr X M(1 2 3 4 5 JP bc1 bc2 bc3 bc4 bc5 bc6 bc7 dt1 dt2 gl oth p) R(C J R) T(c g1 g2 o s tp) W(1 2 3 4)
Windows XP pgo	B
Windows XP debug	B Cpp Jit Mn Wr X M(1 2 3 4 5 JP bc1 bc2 bc3 bc4 bc5 bc6 bc7 dt1 dt2 dt3 dt4 gl oth p) R(C J R) W(1 2 3 4 5 6 7 8)
Windows 7 opt	Cpp Jit Mn Wr X M(1 2 3 4 5 JP bc1 bc2 bc3 bc4 bc5 bc6 bc7 dt1 dt2 gl oth p) M-e10s(bc1 bc2 bc3 bc4 bc5 bc6 bc7) R(C J R) T(c d g1 g2 o s tp x) T-e10s(c d g1 g2 o s tp) W(1 2 3 4)
Windows 7 debug	Cpp Jit Mn Wr X M(1 2 3 4 5 JP bc1 bc2 bc3 bc4 bc5 bc6 bc7 dt1 dt2 dt3 dt4 gl oth p) R(C J R) W(1 2 3 4 5 6 7 8)
Windows 8 x64 opt	B Cpp Jit Mn Wr X M(1 2 3 4 5 JP bc1 bc2 bc3 bc4 bc5 bc6 bc7 dt1 dt2 gl oth p) M-e10s(bc1 bc2 bc3 bc4 bc5 bc6 bc7) R(C J R Ru) T(c d g1 g2 o s tp) T-e10s(c d g1 g2 o s tp) W(1 2 3 4)
Windows 8 x64 pgo	B Cpp Jit Mn Wr X M(1 2 3 4 5 JP bc1 bc2 bc3 bc4 bc5 bc6 bc7 dt1 dt2 gl oth p) M-e10s(bc1 bc2 bc3 bc4 bc5 bc6 bc7) R(C J R Ru) T(c d g1 g2 o s tp) T-e10s(c d g1 g2 o s tp) W(1 2 3 4)
Windows 8 x64 debug	B Cpp Jit Mn Wr X M(1 2 3 4 5 JP bc1 bc2 bc3 bc4 bc5 bc6 bc7 dt1 dt2 dt3 dt4 gl oth p) R(C J R) W(1 2 3 4 5 6 7 8)
Android 2.3 API9 opt	B X1 X2 X3 M(1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 c gl1 gl2 gl3 gl4 rc1 rc2 rc3 rc4) R(C1 C2 J1 J2 J3 J4 J5 J6 R1 R2 R3 R4 R5 R6 R7 R8 R9 R10 R11 R12 R13 R14 R15 R16)
Android 2.3 API9 debug	B
Android 4.0 API11+ opt	B T(rck2 s tpn)
Android 4.0 API11+ debug	B R(C C J1 J1 J2 J3 R1 R2 R3 R4 R5 R6 R7 R8)
Android 4.2 x86 opt	B Sets(S4)
Android 4.3 API11+ opt	Cpp X1 X2 X3 M(1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 c gl1 gl2 gl3 gl4 rc1 rc2 rc3 rc4) R(C1 C2 J1 J2 J3 J4 J5 J6 R1 R2 R3 R4 R5 R6 R7 R8 R9 R10 R11 R12 R13 R14 R15 R16)

Intermittent Failures

Tips and Tricks

Going Backwards

```
(rr) reverse-step
(rr) reverse-next
(rr) reverse-finish
```

```
1991 JSObject*
1992 JSStructuredCloneReader::readSavedFrame(uint32_t principalsTag)
1993 {
1994     RootedSavedFrame savedFrame(context(), SavedFrame::create(context()));
1995     if (!savedFrame)
1996         return nullptr;
1997
1998     JSPrincipals* principals;
1999     if (principalsTag == SCTAG_JSPRINCIPALS) {
2000         if (!context()->runtime()->readPrincipals) {
2001             JS_ReportErrorNumber(context(), GetErrorMessage, nullptr,
2002                                  JSMSG_SC_UNSUPPORTED_TYPE);
2003             return nullptr;
2004         }
2005
2006         if (!context()->runtime()->readPrincipals(context(), this, &principals))
2007             return nullptr;
2008     } else if (principalsTag == SCTAG_RECONSTRUCTED_SAVED_FRAME_PRINCIPALS_IS_SYSTEM) {
2009         principals = &ReconstructedSavedFramePrincipals::IsSystem;
2010         principals->refcount++;
2011     } else if (principalsTag == SCTAG_RECONSTRUCTED_SAVED_FRAME_PRINCIPALS_IS_NOT_SYSTEM) {
2012         principals = &ReconstructedSavedFramePrincipals::IsNotSystem;
2013         principals->refcount++;
2014     } else if (principalsTag == SCTAG_NULL_JSPRINCIPALS) {
2015         principals = nullptr;
2016     } else {
2017         JS_ReportErrorNumber(context(), GetErrorMessage, nullptr,
2018                              JSMSG_SC_BAD_SERIALIZED_DATA, "bad SavedFrame principals");
2019         return nullptr;
2020     }
2021     savedFrame->initPrincipalsAlreadyHeld(principals);
2022
2023     RootedValue source(context());
2024     if (!startRead(&source) || !source.isString())
2025         return nullptr;
2026     auto atomSource = AtomizeString(context(), source.toString());
2027     if (!atomSource)
2028         return nullptr;
2029     savedFrame->initSource(atomSource);
2030
2031     RootedValue lineVal(context());
2032     uint32_t line;
2033     if (!startRead(&lineVal) || !lineVal.isNumber() || !ToUint32(context(), lineVal, &line))
2034         return nullptr;
2035     savedFrame->initLine(line);
2036
2037     RootedValue columnVal(context());
2038     uint32_t column;
2039     if (!startRead(&columnVal) || !columnVal.isNumber() || !ToUint32(context(), columnVal, &column))
2040         return nullptr;
2041     savedFrame->initColumn(column);
2042
2043     RootedValue name(context());
2044     if (!startRead(&name) || !(name.isString() || name.isNull()))
2045         return nullptr;
2046     JSAtom* atomName = nullptr;
2047     if (name.isString()) {
2048         atomName = AtomizeString(context(), name.toString());
2049         if (!atomName)
2050             return nullptr;
2051     }
2052     savedFrame->initFunctionDisplayName(atomName);
2053
2054     RootedValue cause(context());
2055     if (!startRead(&cause) || !(cause.isString() || cause.isNull()))
2056         return nullptr;
2057     JSAtom* atomCause = nullptr;
2058     if (cause.isString()) {
2059         atomCause = AtomizeString(context(), cause.toString());
2060         if (!atomCause)
2061             return nullptr;
2062     }
2063     savedFrame->initAsyncCause(atomCause);
2064
2065     return savedFrame;
2066 }
```

How did this get here??

```
(rr) watch *(void**) &thing->member

(rr) reverse-continue
```

Fin.

http://rr-project.org/

@fitzgen