Firefox

Rethinking Firefox I/O

(The 2020 Edition)

Keefer Rourke
[:krourke]

20.08.25 Firefox Performance Engineering

Firefox is huge and complicated.

A modern browser is a heck of a lot like an operation system.

Let’s change the way we do File I/O.

File I/O in Firefox: A history

1996. NetScape Portable Runtime (C)

Cross-platform I/O library written in C. Provides abstractions for differences in platform calls and structures on Windows and Unixes

2002. XPCOM nsIFile (C++/XPIDL)

Built on top of NSPR, provides object-oriented interfaces for **blocking** file I/O. Exposed to privileged JavaScript via **XPIDL**.

2010. OS.File (JavaScript/C++)

What we use now. Implemented mostly using JavaScript service workers. Provides **asynchronous**/off-main-thread **file I/O** functionality.

2020. IOUtils (C++/Web IDL)

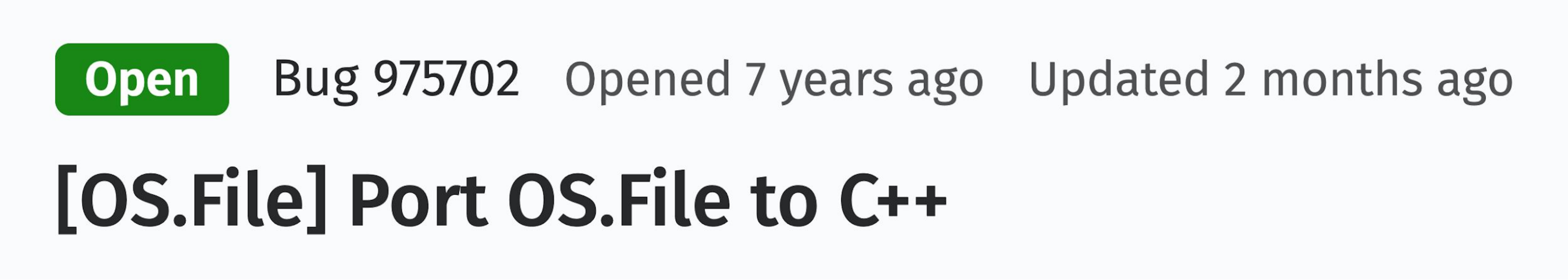
What we'll use going forward. Implemented entirely in C++, and exposed to Firefox chrome code via Web IDL.

Need to do anything with files?

Going forward

We’re using IOUtils from now on!

How did we get here?



Open Bug 975702 Opened 7 years ago Updated 2 months ago

[OS.File] Port OS.File to C++

OS.File's Implementation

Some important numbers (Excluding OS.Path* and tests)

~1500

Lines of C++

~7250

Lines of JavaScript

16

Files (and 2 implementations)

OS.File common usage

Before I started to port OS.File to C++, I did a pretty thorough analysis of how it’s used.

The demand for a new API largely consisted of a namespace for static methods.

```
read
writeAtomic
stat
move
copy
remove
touch
 ...
```

Goals for the API

- Dead easy to use!
- Safe, correct, and consistent across platforms!
- No surprises!
- Support idiomatic JavaScript
- Use a simple implementation built around native promises

Why Yet Another File API?

- The OS.File API is just fine, but the JS implementation has problems
- An opportunity to prune the unused parts of the OS.File interface
- A C++ implementation means
  - Less code
  - More memory savings
  - Less disk I/O
  - A faster Firefox :)

IOUtils

- Provides a non-blocking API to privileged (ChromeOnly) JavaScript
- Uses background thread I/O
- Works on all supported platforms
- Has only one implementation

~1000	0	3
Lines of C++	Lines of JavaScript	Files

Performance

Do less work, and spend less time waiting around.

Doing extra I/O sucks

Most file I/O in the Firefox front-end is done via OS.File

- OS.File is implemented in JS
- JS modules are files on disk

So to do any I/O operation with OS.File

- A C++ process has to read the JS module
- SpiderMonkey has to interpret it
- It has to be kept in memory
- All this has to happen for every process

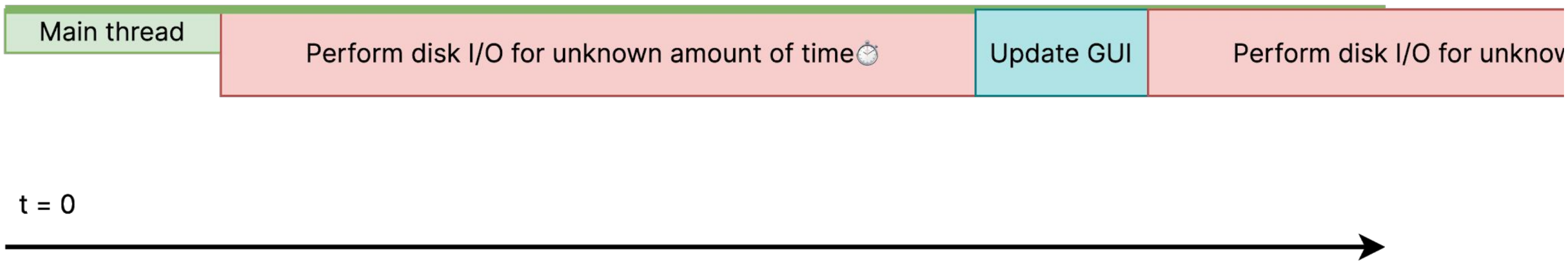
Project Fission will create a lot more processes

- Reducing work during process creation is super important
- IOUtils is native code that comes for free with every process! 😁

Waiting for I/O sucks

- When a thread requests disk I/O, the thread is stuck waiting for results
- Most mobile and desktop apps manage the GUI on the main-thread
  - Including Firefox :)
- I/O bound operations can make apps feel super slow

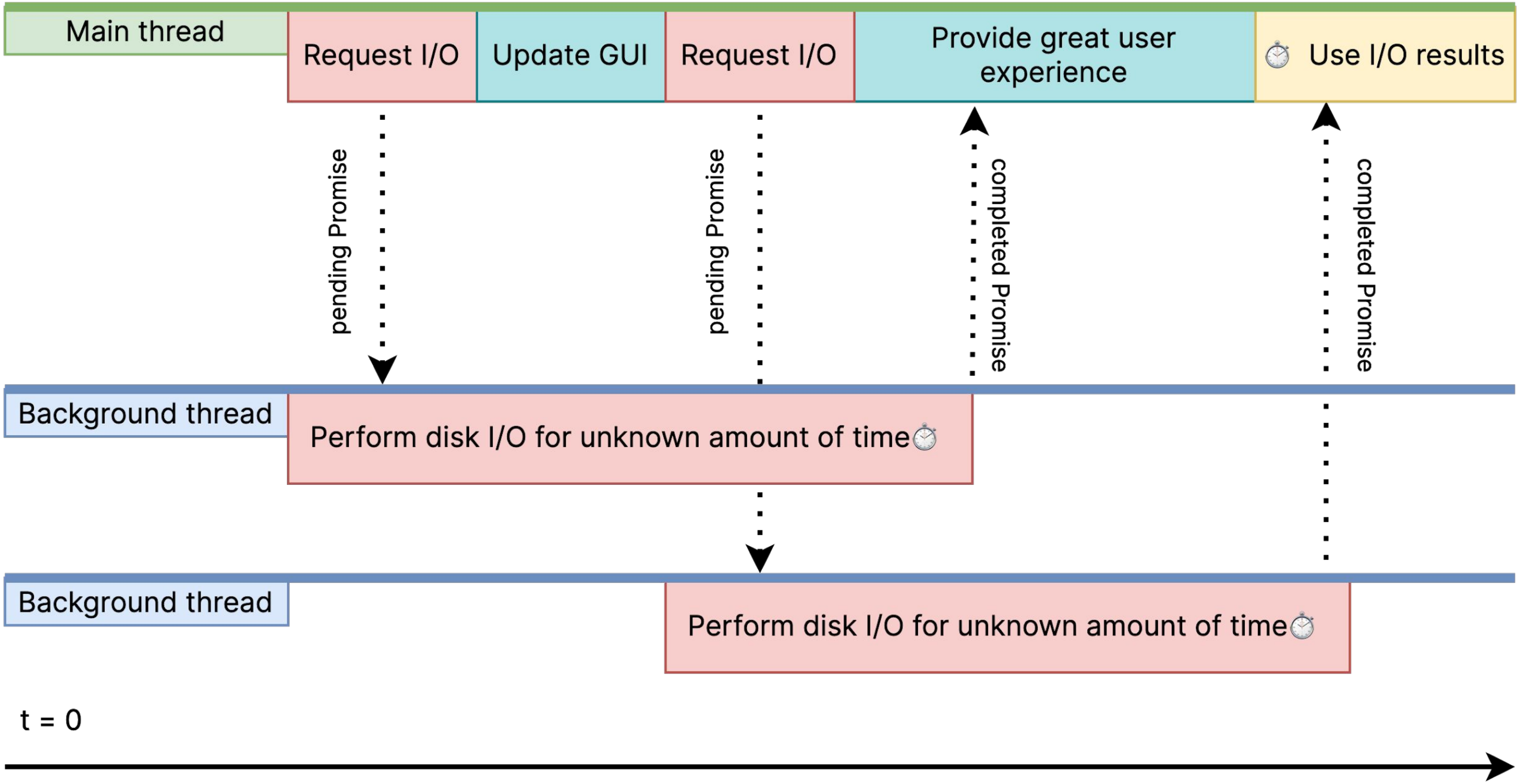
Synchronous I/O



Main thread
Perform disk I/O for unknown amount of time⏱
Update GUI
Perform disk I/O for unknov
t = 0

⏱ = suspended/blocked thread

Asynchronous I/O



What’s next?

Future work

- Consider a blocking version of the API for Rust consumers (bug)
- Consider adding streaming support
  - This could use the W3C streams API
- Stop using OS.File!

Kudos to these folks

Couldn't have got this far without you :)

Kim [:kmoir]
Manager

Barret [:barret]
Mentor

Gijs [:Gijs]
Reviewer

Olli [:smaug]
Reviewer

Some more shout outs

Thanks for helping me with random bugs, giving me great advice, and otherwise making this summer great!

Emma [:emalysz]   **Nika [:nika]**   **Nathan [:froydnj]**   **Anny [:annygakh]**

Benjamin [:b4hand]   **Botond [:botond]**   **David [:Yoric]**

And a big round of applause to these great teams of people :)

The University Team   **The Perf Team**   **Summer 2020 Interns**

Questions?