RFC 8089 : The "file" URI Scheme

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 20 février 2017

Date de publication du RFC : Février 2017

https://www.bortzmeyer.org/8089.html

Vous connaissez le plan d'URI `file:`, qui indique que la ressource se trouve sur le système de fichiers local ? (Par exemple, ce fichier que je suis en train d'éditer est `file:///home/stephane/Blog/RFC/8089.xml`.) Ce plan avait été défini très brièvement dans le RFC 1738 [1] (section 3.10). Tellement brièvement qu'il y manquait pas mal de choses. Ce nouveau RFC remplace cette partie du RFC 1738 et fournit cette fois une description complète du plan `file:`. Ce n'était pas une tâche facile car les différents systèmes de fichiers ont des syntaxes et des comportements très différents. Le RFC lui-même est très court, mais il comporte plusieurs annexes plus longues, discutant de points spécifiques parfois assez tordus.

Donc, d'abord, la définition (section 1 de notre RFC) : un fichier est un objet qui se trouve rangé dans un environnement structuré, qui fournit notamment un système de nommage, environnement qu'on nomme le système de fichiers. (Et le fichier contient des données mais ce point n'est pas crucial pour les URI et n'est donc pas mentionné.) Ensuite, les URI : ce sont les identificateurs standard du Web. Leur syntaxe générique est définie dans le RFC 3986 et ce nouveau RFC ne fait donc que spécialiser le RFC 3986. Normalement, ce RFC est parfaitement compatible avec l'ancienne syntaxe, celle du RFC 1738 mais, en pratique, comme l'ancienne définition était vraiment trop vague, il y aura forcément quelques différences. (L'annexe E donne quelques exemples de pratiques utilisées dans la nature et qui ne sont pas strictement alignées sur les nouvelles règles. Elle cite par exemple l'ajout d'un nom d'utilisateur dans l'URI. Un exemple des problèmes que ces différences posent aux navigateurs est bien expliqué dans cet article de Microsoft <http://blogs.msdn.com/b/ie/archive/2006/12/06/file-uris-in-windows.aspx>.)

Les URI `file:` ne supposent pas l'utilisation d'un protocole particulier, ni d'un type de média particulier.

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc1738.txt`

Ce plan d'URI désigne des « fichiers locaux ». Un fichier local est accessible juste à partir de son nom, sans autre information (par exemple sans utiliser une adresse réseau explicite). Mais, en pratique (section 1.1), il peut être physiquement sur une autre machine, grâce à des techniques comme NFS ou SMB.

La syntaxe de ces URI figure en section 2 de notre RFC, formalisée en ABNF (RFC 5234). S'appuyant sur la syntaxe générique du RFC 3986, elle diffère légèrement de celle du RFC 1738 (l'annexe A liste les différences). Le plan `file:` est référencé dans le registre des plans d'URI <https://www.iana.org/assignments/uri-schemes/uri-schemes.xml>. Je vous laisse découvrir sa grammaire dans le RFC, je donne juste des exemples qui illustrent certains points de la syntaxe :

— Commençons par un URI banal : `file:///tmp/toto.txt`. Il désigne le fichier local `/tmp/toto.txt` de l'ordinateur sur lequel on travaille. La syntaxe du nom de fichier est celle d'Unix, même si ledit ordinateur n'utilise pas Unix. Ainsi, le fichier `c:\machin\truc` sur une machine Windows sera quand même `file:///c:/machin/truc` (il existe plein de variantes non-standard, voir l'annexe E, et l'article cité plus haut <https://blogs.msdn.microsoft.com/ie/2006/12/06/file-uris-in-windows/>, sur les problèmes que cela pose). Sur VMS, `DISK1:[CS.JANE]PAPER.PS` deviendra `file:///disk1/cs/jane/paper.ps` (cf. annexe D).
— Le composant après les trois barres obliques doit être un chemin absolu dans le système de fichiers de la machine. Cela a l'air simple mais la notion de « chemin absolu » ne l'est pas, et l'annexe D cite quelques surprises possibles (comme le tilde de certains shells Unix).
— Après les deux premières barres obliques, il y a normalement un champ nommé « Autorité » (en pratique un nom de domaine), qui est optionnel. Pour les URI `file:`, on peut mettre dans ce champ `localhost`, voire n'importe quel nom qui désigne la machine locale (je ne suis pas sûr de l'intérêt que cela présente, mais c'est la norme qui, il est vrai, déconseille cet usage). Donc, l'URI cité au début aurait pu (mais ce n'est pas recommandé) être `file://localhost/tmp/toto.txt`. (Voir aussi la section 3 du RFC.)
— Si on ne met pas le nom de domaine, les deux premières barres obliques sont facultatives (c'est une nouveauté de notre RFC, par rapport au RFC 1738) et `file:/tmp/toto.txt` est donc légal.
— Certains systèmes de fichiers sont sensibles à la casse et il faut donc faire attention, en manipulant les URI, à ne pas changer la casse. `file:///c:/machin/truc` et `file:///c:/Machin/TRUC` sont deux URI différents même si on sait bien que, sur une machine Windows, ils désigneront le même fichier.

Que peut-on faire avec un fichier ? Plein de choses (l'ouvrir, lire les données, le détruire…La norme POSIX peut donner des idées à ce sujet.) Le plan d'URI `file:` ne limite pas les opérations possibles.

Évidemment, l'encodage des caractères utilisé va faire des histoires, puisqu'il varie d'une machine à l'autre. C'est parfois UTF-8, parfois un autre encodage et, parfois, le système de fichiers ne définit rien, le nom est juste une suite d'octets, qui devra être interprétée par les applications utilisées (c'est le cas d'Unix). Notre RFC (section 4) recommande bien sûr d'utiliser UTF-8, avant l'optionelle transformation pour cent (RFC 3986, section 2.5). Ainsi, le fichier `/home/stéphane/café.txt` aura l'URI `file:/home/st%C3%A9phane/caf%C3%A9.txt`, quel qu'ait été son encodage sur la machine. Au passage, j'ai essayé avec curl et `file:///tmp/café.txt`, `file:/tmp/café.txt`, `file:/tmp/caf%C3%A9.txt`, `file://localhost/tmp/caf%C3%A9.txt` et même `file://mon.adresse.ip.publique/tmp/caf%C3%A9.` marchent tous.

Et la sécurité ? Toucher aux fichiers peut évidemment avoir des tas de conséquences néfastes. Par exemple, si l'utilisateur charge le fichier `file:///home/michu/foobar.html`, aura-t-il la même origine (au sens de la sécurité du Web) que `file:///tmp/youpi.html` ? Après tout, ils viennent du même domaine (le domaine vide, donc la machine locale). Le RFC note qu'au contraire l'option la plus sûre est de considérer que chaque fichier est sa propre origine (RFC 6454).

Autre question de sécurité rigolote, les systèmes de fichiers ont en général des caractères spéciaux (comme la barre oblique ou un couple de points pour Unix). Accéder bêtement à un fichier en passant

juste le nom au système de fichiers peut soulever des problèmes de sécurité (c'est évidemment encore pire si on passe ces noms à des interpréteurs comme le shell, qui rajoutent leur propre liste de caractères spéciaux). Le RFC ne spécifie pas de liste de caractères « dangereux » car tout nouveau système de fichiers peut l'agrandir. C'est aux programmeurs qui font les logiciels de faire attention, pour le système d'exploitation pour lequel ils travaillent. (Un problème du même ordre existe pour les noms de fichiers spéciaux, comme `/dev/zero` sur Unix ou `aux` et `lpt` sur Windows.)

Une mauvaise gestion de la sensibilité à la casse ou de l'encodage des caractères peut aussi poser des problèmes de sécurité (voir par exemple le rapport technique UAX #15 <http://www.unicode.org/reports/tr15/tr15-44.html> d'Unicode.)

Notons qu'il existe d'autres définitions possibles d'un URI `file:` (annexe C de notre RFC). Par exemple, le WhatWG maintient une liste des plans d'URI <http://url.spec.whatwg.org/>, non synchronisée avec celle « officielle » <https://www.iana.org/assignments/uri-schemes/uri-schemes.xml>, et dont l'existence a fait pas mal de remous à l'IETF, certains se demandant s'il fallait quand même publier ce RFC, au risque d'avoir des définitions contradictoires (cela a sérieusement retardé la sortie du RFC). En gros, l'IETF se concentre plutôt sur la syntaxe, et le WhatWG sur le comportement des navigateurs (rappelez-vous que les URI ne sont pas utilisés que par des navigateurs…). Il y a aussi les définitions Microsoft comme UNC <http://msdn.microsoft.com/en-us/library/gg465305.aspx> ou leurs règles sur les noms de fichier <https://msdn.microsoft.com/en-us/library/windows/desktop/aa365247(v=vs.85).aspx>.

Et, pour finir, je vous recommande cet autre article de Microsoft <https://blogs.msdn.microsoft.com/freeassociations/2005/05/19/the-bizarre-and-unhappy-story-of-file-urls/> sur l'évolution du traitement des URI dans IE.