RFC 8792 : Handling Long Lines in Content of Internet-Drafts and RFCs

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 3 juillet 2020

Date de publication du RFC : Juin 2020

https://www.bortzmeyer.org/8792.html

Le sujet peut sembler un point de détail sans importance mais, dans un RFC, tout compte. Que serait une norme si les exemples de code et les descriptions en langage formel avaient la moindre ambiguïté ? C'est le cas par exemple des lignes trop longues : comment les couper pour qu'on comprenne bien ce qui s'est passé ? Ce RFC propose deux méthodes.

Le cas principal est celui des modules YANG (RFC 7950 [1]) et c'est pour cela que ce RFC sort du groupe de travail netmod <https://datatracker.ietf.org/wg/netmod/>. Mais le problème concerne en fait tous les cas où on veut mettre dans un RFC du texte dans un langage formel et que les lignes dépassent ce qui est généralement considéré comme acceptable. Notre RFC formalise deux solutions, toutes les deux traditionnelles et déjà souvent utilisées, mettre une barre inverse après les lignes coupées, ou bien mettre cette barre inverse et une autre au début de la ligne suivante. Donc, dans le premier cas, le texte :

```
Il n'est pas vraiment très long, mais c'est un exemple.
```

Ce texte, coupé à quarante-cinq caractères, donnerait :

```
Il n'est pas vraiment très long, mais c'est \
un exemple.
```

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc7950.txt

Ou bien :

```
Il n'est pas vraiment très long, mais c'est \
       un exemple.
```

Et, avec la seconde solution :

```
Il n'est pas vraiment très long, mais c'est \
\un exemple.
```

Si la forme officielle des RFC n'est plus le texte brut depuis le RFC 7990, il y a toujours une version en texte brut produite, suivant le format décrit dans le RFC 7994. Il impose notamment des lignes de moins de 72 caractères. Or, les exemples de code, ou les textes décrits en un langage formel (modules YANG, ABNF, ou ASN.1, par exemple), dépassent facilement cette limite. Les outils actuels comme xml2rfc `<https://pypi.org/project/xml2rfc/>` ne gèrent pas vraiment ce cas, se contentant d'afficher un avertissement, et laissant l'auteur corriger lui-même, d'une manière non officiellement standardisée.

Désormais, c'est fait, il existe un tel standard, ce nouveau RFC, standard qui permettra notamment de bien traiter les textes contenus dans l'élement `<sourcecode>` (RFC 7991, section 2.48) du source du RFC. Naturellement, rien n'interdit d'utiliser cette norme pour autre chose que des RFC (section 2 du RFC).

Les deux stratégies possibles, très proches, sont « une seule barre inverse » et « deux barres inverses ». Dans les deux cas, un en-tête spécial est mis au début, indiquant quelle stratégie a été utilisée, pour qu'il soit possible de reconstituer le fichier original.

Dans le premier cas, la stratégie à une seule barre inverse (section 7 du RFC), une ligne est considérée pliée si elle se termine par une barre inverse. La ligne suivante (moins les éventuels espaces en début de ligne) est sa suite. Dans le deuxième cas, la stratégie à deux barres inverses (section 8), une autre barre inverse apparait au début des lignes de continuation.

Différence pratique entre les deux stratégies ? Avec la première (une seule barre inverse), le résultat est plus lisible, moins encombré de caractères supplémentaires. Par contre, comme les espaces au début de la ligne de continuation sont ignorés, cette stratégie va échouer pour des textes comportant beaucoup d'espaces consécutifs. La deuxième est plus robuste. La recommandation du RFC est donc d'essayer d'abord la première stratégie, qui produit des résultats plus jolis, puis, si elle échoue, la deuxième. L'outil `rfcfold`, présenté plus loin, fait exactement cela par défaut.

La section 4 rappellait le cahier des charges des solutions adoptées :

— Repli des lignes automatisable (cf. le script `rfcfold` présenté à la fin), notamment pour le cas où un RFC ou autre document est automatiquement produit à partir de sources extérieures,
— Reconstruction automatisable du fichier original, pour qu'on puisse réutiliser ce qu'on a trouvé dans un RFC en texte brut, ce qui est déjà utilisé par les outils de soumission d'"*Internet-Drafts*", qui extraient le YANG automatiquement et le valident, pratique qui pourrait être étendu à XML et JSON, et qui assure une meilleure qualité des documents, dès la soumission.

En revanche, la section 5 liste les « non-buts » : il n'est pas prévu que les solutions fonctionnent pour l'art ASCII. Et il n'est pas prévu de gérer les coupures spécifiques à un format de fichier. Les stratégies décrites dans ce RFC sont génériques. Au contraire, des outils comme pyang <https://pypi.org/project/pyang/> ou yanglint <https://github.com/CESNET/libyang#yanglint> connaissent la syntaxe de YANG et peuvent plier les lignes d'un module YANG plus intelligemment (cf. aussi RFC 8340).

L'annexe A du RFC contient un script bash qui met en œuvre les deux stratégies (le script). Ici, je pars d'un fichier JSON compact (au format du RFC 8785). D'abord, première stratégie (notez la ligne de documentation ajoutée au début) :

```
% ./rfcfold.bash -s 1 -i /tmp/i.txt -o /tmp/t.txt && cat /tmp/t.txt
========== NOTE: '\' line wrapping per RFC 8792 ===========

[{"af":6,"avg":23.4686283333,"dst_addr":"2001:67c:2218:e::51:41","ds\
t_name":"2001:67c:2218:e::51:41","dup":0,"from":"2a01:4240:5f52:bbc4\
::ba3","fw":5020,"group_id":26102101,"lts":30,"max":23.639063,"min":\
23.285885,"msm_id":26102101,"msm_name":"Ping","mver":"2.2.1","prb_id\
":1000276,"proto":"ICMP","rcvd":3,"result":[{"rtt":23.480937},{"rtt"\
:23.639063},{"rtt":23.285885}],"sent":3,"size":64,"src_addr":"2a01:4\
240:5f52:bbc4::ba3","step":null,"stored_timestamp":1593694541,"times\
tamp":1593694538,"ttl":52,"type":"ping"},{"af":6,"avg":65.9926666667\
...
```

Et avec la deuxième stratégie :

```
% ./rfcfold.bash -s 2 -i /tmp/i.txt -o /tmp/t.txt && cat /tmp/t.txt|more
========== NOTE: '\\' line wrapping per RFC 8792 ==========

[{"af":6,"avg":23.4686283333,"dst_addr":"2001:67c:2218:e::51:41","ds\
\t_name":"2001:67c:2218:e::51:41","dup":0,"from":"2a01:4240:5f52:bbc\
\4::ba3","fw":5020,"group_id":26102101,"lts":30,"max":23.639063,"min\
\":23.285885,"msm_id":26102101,"msm_name":"Ping","mver":"2.2.1","prb\
\_id":1000276,"proto":"ICMP","rcvd":3,"result":[{"rtt":23.480937},{"\
\rtt":23.639063},{"rtt":23.285885}],"sent":3,"size":64,"src_addr":"2\
\a01:4240:5f52:bbc4::ba3","step":null,"stored_timestamp":1593694541,\
\"timestamp":1593694538,"ttl":52,"type":"ping"},{"af":6,"avg":65.992\
```

L'option `-r` du script permet d'inverser, c'est-à-dire de reconstituer le fichier original à partir du fichier aux lignes pliées.

L'outil `rfcfold` n'a pas de connaissance des formats de fichiers et coupe donc brutalement, y compris au milieu d'un nom de variable. Un outil plus intelligent (section 9.3) pourrait couper au bon endroit, par exemple, pour XML, juste avant le début d'un élément XML.