RFC 4234 : Augmented BNF for Syntax Specifications: ABNF

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 7 septembre 2006. Dernière mise à jour le 1 février 2008

Date de publication du RFC : Octobre 2005

https://www.bortzmeyer.org/4234.html

Ce RFC, désormais remplacé par le RFC 5234 [1], fait partie des RFC ancillaires, qui ne spécifient pas directement un protocole IETF mais fournissent des outils pour les "vrais" RFC. En l'occurrence, il normalise le mini-language pour écrire des grammaires.

Beaucoup de RFC doivent spécifier un langage, en général assez simple (jamais de la taille d'un langage de programmation) mais néanmoins suffisamment important pour que les descriptions informelles du langage soient risquées. Depuis longtemps, on utilise en informatique des notations dérivées du langage BNF pour spécifier formellement un langage. Le problème est qu'il existe plusieurs dialectes de BNF (comme EBNF) et que les RFC ont besoin d'une référence stable. D'où ce RFC, qui succède au fameux RFC 2234 (avec très peu de changements), et qui a lui-même été remplacé par le RFC 5234. Il décrit **ABNF**, le dialecte IETF de BNF. Classique sur beaucoup de points, ce dialecte a quand même quelques variations, issues d'une histoire très ancienne. Par exemple, le signe — pour le choix est remplacé par /.

Quelques outils sont disponibles <http://tools.ietf.org/inventory/verif-tools> pour aider les auteurs de grammaires. Mais je trouve que c'est encore insuffisant. S'il existe deux vérificateurs (qui peuvent tester qu'une grammaire est cohérente), il n'existe guère de générateurs d'analyseurs syntaxiques. En revanche, à des fins de test <https://www.bortzmeyer.org/test-logiciel.html>, il existe un programme, Eustathius <https://www.bortzmeyer.org/eustathius-test-grammars.html>, qui génère automatiquement des exemples à partir d'une grammaire. Vous trouverez de nombreux exemples de grammaires ABNF dans les sources d'Eustathius.

À titre d'exemple, voici la spécification de SPF (décrit dans le RFC 4408) en ABNF :

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc5234.txt

```
record           = version terms *SP
version          = "v=spf1"

terms            = *( 1*SP ( directive / modifier ) )

directive        = [ qualifier ] mechanism
qualifier        = "+" / "-" / "?" / "~"
mechanism        = ( all / include
                   / A / MX / PTR / IP4 / IP6 / exists )

all              = "all"
include          = "include"  ":" domain-spec
A                = "a"      [ ":" domain-spec ] [ dual-cidr-length ]
MX               = "mx"     [ ":" domain-spec ] [ dual-cidr-length ]
PTR              = "ptr"    [ ":" domain-spec ]
IP4              = "ip4"      ":" ip4-network   [ ip4-cidr-length ]
IP6              = "ip6"      ":" ip6-network   [ ip6-cidr-length ]
exists           = "exists"   ":" domain-spec

modifier         = redirect / explanation / unknown-modifier
redirect         = "redirect" "=" domain-spec
explanation      = "exp" "=" domain-spec
unknown-modifier = name "=" macro-string

ip4-cidr-length  = "/" 1*DIGIT
ip6-cidr-length  = "/" 1*DIGIT
dual-cidr-length = [ ip4-cidr-length ] [ "/" ip6-cidr-length ]

ip4-network      = qnum "." qnum "." qnum "." qnum
qnum             = DIGIT                 ; 0-9
                   / %x31-39 DIGIT       ; 10-99
                   / "1" 2DIGIT          ; 100-199
                   / "2" %x30-34 DIGIT   ; 200-249
                   / "25" %x30-35        ; 250-255
          ; conventional dotted quad notation.  e.g., 192.0.2.0
ip6-network      = <as per [RFC 3513], section 2.2>
          ; e.g., 2001:DB8::CD30

domain-spec      = macro-string domain-end
domain-end       = ( "." toplabel [ "." ] ) / macro-expand
toplabel         = ( *alphanum ALPHA *alphanum ) /
                   ( 1*alphanum "-" *( alphanum / "-" ) alphanum )
                   ; LDH rule plus additional TLD restrictions
                   ; (see [RFC3696], Section 2)

alphanum         = ALPHA / DIGIT

explain-string   = *( macro-string / SP )

macro-string     = *( macro-expand / macro-literal )
macro-expand     = ( "%{" macro-letter transformers *delimiter "}" )
                   / "%%" / "%_" / "%-"
macro-literal    = %x21-24 / %x26-7E
                   ; visible characters except "%"
macro-letter     = "s" / "l" / "o" / "d" / "i" / "p" / "h" /
                   "c" / "r" / "t"
transformers     = *DIGIT [ "r" ]
delimiter        = "." / "-" / "+" / "," / "/" / "_" / "="

name             = ALPHA *( ALPHA / DIGIT / "-" / "_" / "." )

header-field     = "Received-SPF:" [CFWS] result FWS [comment FWS]
                   [ key-value-list ] CRLF

result           = "Pass" / "Fail" / "SoftFail" / "Neutral" /
                   "None" / "TempError" / "PermError"

key-value-list   = key-value-pair *( ";" [CFWS] key-value-pair )
```

```
                  [";"]

key-value-pair   = key [CFWS] "=" ( dot-atom / quoted-string )

key              = "client-ip" / "envelope-from" / "helo" /
                   "problem" / "receiver" / "identity" /
                    mechanism / "x-" name / name

identity         = "mailfrom"   ; for the "MAIL FROM" identity
                   / "helo"     ; for the "HELO" identity
                   / name       ; other identities

dot-atom         = <unquoted word as per [RFC2822]>
quoted-string    = <quoted string as per [RFC2822]>
comment          = <comment string as per [RFC2822]>
CFWS             = <comment or folding white space as per [RFC2822]>
FWS              = <folding white space as per [RFC2822]>
CRLF             = <standard end-of-line token as per [RFC2822]>
```