http://clx.asso.fr/spip/?Le-sendmail-cf

SENDMAIL

Le sendmail.cf

- Documentations - Technique -

Date de mise en ligne : mercredi 24 avril 2002

Copyright © Club LinuX Nord-Pas de Calais - Tous droits réservés

Le fichier sendmail.cf fait peur. Et à juste titre. Aujourd'hui, on ne l'utilise presque plus, on préfÚre passer par *m4* et les fichiers *config.mc*, qui facilitent grandement la tâche.
Cependant, dans certains cas, il reste utile de savoir comment fonctionne ce fichier, et quels avantages on peut tirer de son paramétrage.

Prérequis

Ce document suppose que vous savez déjà ce qu'est un email, ce qu'est un serveur de messagerie, que vous avez une connexion Internet, et que vous connaissez au moins quelques points concernant le protocole SMTP, l'échange des emails, et le format des emails. Voir la norme (indigeste) RFC 2822.
Il suppose également que vous avez déjà utilisé Sendmail, et que vous savez le paramétrer avec les macros m4. Voir à ce propos l'excellente documentation d'Eric JACOBONI.

Le fichier sendmail.cf est découpé en trois parties principales.

Les variables d'"environnement"

Tout en haut, les définitions de variables (nommées macros) simples, relatives au fonctionnement du site sur lequel est installé sendmail sont consignées. Les plus utiles sont

- `Cw` : indique le nom sous lequel est connu le serveur, et pour lequel il acceptera le courrier (fonctionne également si le serveur possède plusieurs noms) ;
- `DS` : donne le relais SMTP éventuel à utiliser ;
- `CG` : précise pour quel domaine appliquer la transformation d'adresse (genericstable) ;
- `CM` son pendant pour le *masquerading*, c'est à dire la transformation du domaine "local" en domaine connu sur Internet ;
- `DM` : définit le domaine (connu sur internet) qui remplacera le domaine local lors de la phase de "masquerading".

Les options de fonctionnement

Au milieu du fichier se trouvent les options de fonctionnement.
Elles commencent par "O".
Les plus utiles sont :

- `MaxMessageSize` : limite la taille des emails envoyés ;
- `HoldExepensive` : permet de "conserver" un mail en attente, si le serveur SMTP se situe derrière un routeur Numéris (ou de type RTC). Ainsi, vous évitez que le routeur ne se connecte à Internet dès qu'il reçoit une demande de connexion émanent de Sendmail ;

Copyright © Club LinuX Nord-Pas de Calais

– `LogLevel` : définit la "verbosité" des fichiers de journalisation (les fameux logs repris ou pas par syslog) ;
– `Timeout.*` : indique les délais d'expiration lors de l'envoi de mails. Les plus utiles sont queuewarn (4h) et queuereturn (3d).
Ici, un avertissement (warning) est déclenché au bout de 4 heures et les mails reviennent au destinataires au bout de 3 jours d'essais infructeux de Sendmail pour transmettre le courrier.
On peut changer ces delais si Sendmail se trouve derrière un modem, et que les connexions ne sont pas régulières... Le warning intervient au bout d'une journée, le return au bout de 7 jours, par exemple...

Après ces options, deux ou trois "champs" qu'il vaut mieux laisser tels quels... Enfin... Moi je n'ai jamais eu besoin d'y toucher ;-)

Les règles - REWRITING RULES

Ah. Nous y voilà. La partie la plus complexe à appréhender, et certainement la plus intéressante.

En fait, l'ensemble de signes cabalistiques que vous retrouvez tout en bas, ce sont des règles de transformation, basées sur des *expressions rationelles [1]*

Les règles de réécriture dans le fichier sendmail.cf sont découpées en plusieurs parties, chaque règle pouvant faire appel à une autre en fonction des besoins.
Les règles par défaut - et leur fonctions - sont les suivantes :
– R3 (point d'entrée obligatoire) ;
– R0 : "livraison" de l'email ;
– R1 : traitement de l'adresse de l'expéditeur ;
– R2 : traitement de l'adresse du destinataire ;
– R4 : point de sortie du traitement sendmail.

Les règles 3 et 4 servent à remettre les champs en forme, respectivement pour un traitement plus facile par sendmail (R3), et pour le remettre comme il faut (R4).
Les règles 1 et 2 peuvent être changées selon les besoins ; elle ou leurs "sous fonctions", *ie.* "sous règles" appelées par ces règles. Ces dernières sont définies par les macro m4 actuellement utilisées, et ont chacune un usage particulier. On citera entre autres :
– S93 et S94 : pour faire le masquerading ;
– S31 et S61 : pour traiter la clause "all_masquerade" ;
– S51 : pour transformer les pseudo-domaines (locaux) en "fullly qualified domain names" que l'on trouve sur internet. [2]

Syntaxe des règles

Une règle commence par `SXX`, où XX est le numéro de la règle.
Dans le fichier sendmail.cf, les commentaires sont marqués par des "#".

Chaque ligne d'une règle commence par un 'R', et est décomposée en deux partie :
– La LHS (Left Hand Side), est à gauche, c'est en fait une expression régulière ;

– La RHS (Right Hand Side), est à droite, c'est la règle de réécriture ;
– Les commentaires ; à l'époque où les sendmail.cf était gravés au marteau et au burin, il fallait bien s'y retrouver.
Les trois parties sont séparées par une ou plusieurs tabulation(s).

Les lignes d'une règle sont traitées de façon séquentielle, du début à la fin de la règle.

Prenons, par exemple, cette ligne, la première de la règle S93 :

```
R$+ < @ $=G . >   $: <$1@$2 > $1 < @ $2 . > @  mark
```

Que cela signifie-t-il ?

Il faut d'abord savoir qu'à cet endroit, les adresses ont déjà été traitées par Sendmail.
Un '.' en bout d'adresse signifie que c'est une adresse absolue (FQDN), c'est à dire adresse email + nom de domaine.
Des espaces ont été introduites entre les signes de l'adresse, c'est à dire que "gaetan@clx" a été converti en "< gaetan @ clx . anet . fr . >"
En d'autres termes, des espaces ont été insérées entre les "tokens" [3]

La LHS

La partie gauche (LHS), l'expression régulière dit ceci :
si ("'un token ou plus' < @ 'une adresse dans $G' . >")
exécuter la RHS

Les "$" dans la LHS sont des opérateurs spéciaux, qui agissent comme les expressions rationelles :
– $* : 0 token ou plus ;
– $+ : 1 token ou plus ;
– $- : exactement un token ;
– $@ : exactement 0 token ;
– $=N : n'importe quel token de la classe N [4]
– $ N : n'importe quel token s'il n'est pas dans la classe N.

Alors, ça paraît simple ? J'explique encore un peu plus ; sinon, passez à la suite :

```
R$+ < @ $=G . >
```

"Un ou plusieurs token(s), un '>', un @, un ou plusieurs token(s) de la classe G [5], un point, un signe '>'
Si cette condition est vérifiée, on effectue le remplacement par la RHS.

La RHS

La partie droite (RHS), la partie remplacement, est assez complexe :

```
R$+ < @ $=G . >   $: <$1@$2 > $1 < @ $2 . > @  mark
```

Ici, la partie gauche commençait avec un "$:", qui est l'un des cinq préfixes possibles pour la RHS. Ils sont :

- `$:` : écrire, ne pas relire cette ligne, puis continuer le traitement jusqu'à la fin de la règle ;
- `$@` : écrire, ne pas relire cette ligne, et arrêter le traitement sans aller jusqu'au bout de la règle ;
- `$>SXX` : écrire, puis passer la main à la règle XX, et continuer le traitement jusqu'à la fin de la règle ;
- `$#` : interrompre le traitement et désigner un mode de "livraison". Typiquement un arrêt sur erreur ;
- rien du tout : écrire, relire cette ligne avant de continuer le traitement jusqu'à la fin de la règle.

On trouve également d'autres opérateurs de remplacement dans la RHS :

- `$XX` : écriture du token XX tel que trouvé par la LHS ;
- `$[ $]` : remplacer le token par son FQDN [6] ;
- `$( $)` : remplacer par le champ correspondant dans la base spécifiée [7].

Reprenons notre règle à la lueur (faible) de ces explications (faibles) :

```
$: <$1@$2 > $1 < @ $2 . > @
```

On ne traite la ligne qu'une seule fois. Après traitement, on passera donc directement à la suivante. Cela dit, si la partie gauche (LHS) correspond, alors on écrit :
"< 'le premier token trouvé'@'le second token trouvé'> 'le premier token trouvé'
< @ 'le seconde token trouvé' .> @"

La partie gauche est déjà réécrite en partie pour la suite du traitement, la partie droite contient encore les tokens qui seront traités par la suite.

Exemple complet

Revoici l'intégralité (du début) de la règle S93 :

Copyright © Club LinuX Nord-Pas de Calais

```
 R$+ < @ $=G . >  $: <$1@$2 > $1 < @ $2 . > @  {déjà traité}
R$+ < @ *LOCAL* > $: <$1@$j > $1 < @ *LOCAL* > @  {adresse email non FQDN (cf. les règles antérieures dans le sendmail.cf)}
R< $+ > $+ < $* > @ $: < $(generics $1 $: $) > $2 < $3 > (appel à la base genercistable}
R< > $+ < @ $+ >  $: < $(generics $1 $: $) > $1 < @ $2 > (idem, mais sans 'commentaire' à l'email}
R< $* @ $* > $* < $* > $@ $>3 $1 @ $2    {On a trouvé un email FQDN; appel à la Règle 3,} R< $+ > $* < $* > $: $>3 $1 @ *LOCAL*    {On a trouvé un email non FQDN; appel à la Règle 3,}
R< > $*   $: $1     {Rien trouvé, on remet comme c'était avant.}
```

La seconde ligne s'applique aux adresses déjà marquées "locales", c'est à dire dont le nom complet (hôte + domaine) n'est pas écrit, par exemple, `gaetan@clx`. Si vous avez activé l'option "always_add_domain" dans le fichier m4, la transformation est déjà faite. Donc cette ligne ne sert pas. Sinon, cela permet de traiter aussi les emails "locaux" ;-)

Les troisième et quatrième lignes remplacent les tokens trouvés en lignes 1 et 2 par leur correspondant dans la base genericstable.

Les cinquième et sixième lignes font appel à la règle 3 pour continuer le traitement.

La septième ligne efface les '< >' posés en première et seconde lignes si les troisèmes à sixièmes n'ont rien donné.

Une remarque s'impose. Dans la mesure où les lignes de remplacement des règles de sendmail sont... des opérations de remplacement, il FAUT
TOUJOURS faire en sorte de conserver TOUTE l'information. C'est pour ça que la septième ligne existe, et que les tokens sont écrits deux fois dans les lignes 1 et 2. Mais vous verrez ça tout seuls si vous lisez un petit peu le fichier sendmail.cf après avoir lu ce texte ;-)

Conclusion

Alors ? Je le reconnais, ce n'est pas évident, et je ne vais pas tout vous expliquer comme ça. Il y a des points qui ne sont pas encore très clairs (hé, je ne m'appelle pas E.ALLMAN, non plus...). Mais à la lueur de ce document, j'espère que vous verrez sendmail d'un autre oeil, et surtout, que vous en aurez moins peur...

---

[1] on dit souvent expressions régulières.
En anglais, on peut dire les deux. En France, on préfèrera la première, par respect pour un long long débat qui a fait souffrir un certain nombre de drosophiles, et par respect pour René COUGNEC qui y a eu le dernier mot, mais qui nous a quitté depuis.

[2] Pour plus de détails sur ces règles et leur fonctions, je ne peux que trop vous conseiller l'excellent livre "SENDMAIL, par Bryan Costales et Eric Allman, ISBN 1-56592-222-0 (pour la 2nde édition), des éditions O'Reilly. Ici, c'est la version anglaise, plus à jour que la version Française. O'Reilly publie régulièrement de nouvelles éditions.

[3] dans l'exemple précédent,
on trouve 4 tokens : gaetan, clx, anet et fr

[4] On définit un classe par CN dans la première partie,
par exemple CM pour le masquerading.

[5] *ie.* contenu(s) dans
le fichier genericstable

[6] Fully Qualified Domain Name. Vous suivez ou quoi ?

[7] Ici,

Copyright © Club LinuX Nord-Pas de Calais

ce sera, dans la base genericstable. Mais on verra ça dans deux paragraphes.

Copyright © Club LinuX Nord-Pas de Calais