Sendmail v8
Kit Jussieu

Pierre David

Novembre 2001

La composition de ce document a été effectuée par un ordinateur avec le système d'exploitation Unix (plus spécifiquement une version du domaine public), en utilisant le logiciel de composition TEX (domaine public). Les figures ont été dessinées sous X-Window (domaine public) avec le logiciel `xfig` (domaine public) et intégrées directement dans le document final. La version HTML a été générée avec LATEX2HTML (domaine public).

La dernière version de ce document est accessible à l'adresse :

`ftp://ftp.jussieu.fr/jussieu/sendmail/kit/`

La version HTML de ce document est accessible à l'adresse :

`http://www-crc.u-strasbg.fr/docs/kit-jussieu/`

Version 5.4, novembre 2001

Table des matières

Introduction

Il est banal, aujourd'hui, de dire que l'Internet connaît un succès foudroyant. Ce succès entraîne dans son sillage (ou est peut-être dû en partie à) la messagerie basée sur le protocole SMTP. L'implémentation la plus répandue de ce protocole est sans conteste le programme `sendmail` conçu par Eric Allman en 1980. Après plusieurs années sans maintenance claire, ayant vu apparaître plusieurs versions dérivées et divergentes, Eric en a repris depuis 1990 la maintenance et l'extension pour prendre en compte les nouveaux besoins exprimés et standardisés, comme MIME et DSN par exemple.

La configuration de `sendmail` (la rédaction du fichier `sendmail.cf`) a toujours été une étape difficile dans la vie d'un administrateur de messagerie. Les difficultés ont sans doute des origines multiples, mais ont généralement en commun la complexité et la diversité des situations que doit gérer `sendmail`.

Une nouvelle évolution s'est dessinée ces dernières années : il n'y a plus guère de configurations de `sendmail` écrites « à la main ». La plupart des configurations sont obtenues à partir de définitions de plus ou moins haut niveau, et le fichier `sendmail.cf` est généré automatiquement. À titre d'exemple, on peut citer l'ensemble de macros `m4` fourni avec les sources de `sendmail` lui-même, ou le Kit de Jussieu décrit ici.

Les premières versions de ce document étaient destinées à servir de support pour des cours organisés par le CRU (Comité Réseau des Universités) et l'UREC (Unité Réseaux du CNRS). Compte-tenu des remarques positives sur son contenu, j'ai choisi de le rendre accessible au plus grand nombre. Il est décomposé en trois parties.

La première décrit la messagerie en général, et ses récentes évolutions. Elle peut être lue par toute personne confrontée à la messagerie, utilisateur comme administrateur.

La seconde partie décrit le programme `sendmail` ainsi que les grandes lignes du fichier de configuration. Cette partie peut être lue par tout administrateur de la messagerie, que le fichier `sendmail.cf` soit généré par le Kit de Jussieu, par les macros `m4` mentionnées ci-dessus, ou même par l'administrateur directement.

La troisième et dernière partie décrit le Kit de Jussieu. Ce kit a été développé en espérant qu'il soit utile, tant pour les sites importants (universitaires ou non) que pour les particuliers raccordés derrière un fournisseur d'accès à l'Internet. Aucune garantie d'aucune sorte ne l'accompagne, bien que les auteurs (Jacky Thibault, Sébastien Vautherot et moi-même) feront tout leur possible pour faciliter son utilisation.

Strasbourg, le 25 novembre 2001.

Chapitre 1

Courrier électronique

Le courrier électronique sur l'Internet est défini par un certain nombre de règles qu'il n'est pas inutile de rappeler.

1.1 Structure d'un système de messagerie

Un système de messagerie Internet contient les composants suivants :

- *agent utilisateur* (ou *user agent*, ou UA).
  Un UA est un programme que l'utilisateur emploie pour composer son message et l'envoyer à l'agent de routage (voir ci-dessous) pour l'injecter dans le système de messagerie.
  Les UA typiques sous Unix vont du plus simple, `mail` à l'interface ligne rudimentaire, aux plus graphiques comme `exmh` avec X-Window et gestion de MIME (voir section 1.8), en passant par les plus perfectionnés comme `mutt`, digne successeur d'`elm`, intégrant des fonctionnalités très avancées.
  Un UA permet également la lecture du courrier. C'est la phase finale. Il y a un UA à chaque extrémité du système de messagerie.
- *agent de routage des messages*
  Un agent de routage reçoit un message. En fonction de l'adresse du destinataire, il décide de faire appel à un agent de transport de messages, dont le but est d'acheminer le message dans la direction du destinataire.
  Les agents de routage sous Unix sont, entre autres, `sendmail`, `smail`, `mmdf` et `postfix`. Toutefois, le plus répandu est incontestablement `sendmail`.
- *agent de transport des messages*
  Un agent de transport reçoit un message et une direction, et l'achemine à l'endroit indiqué. Il faut bien comprendre que l'agent de transport des messages ne prend pas de décision quant au routage. Cette décision lui est indiquée par l'agent de routage qui lui transmet le message.
  Un agent de transport de messages est spécialisé pour un type de transmission. Par exemple, il y aura un agent de transport pour SMTP (le protocole utilisé sur l'Internet), un autre pour la remise physique du message (lorsque le courrier arrive dans la boîte aux lettres du destinataire), un troisième pour l'expédition vers un site UUCP, etc.
  En terminologie `sendmail`, un agent de transport est appelé un *mailer* (bien qu'en terminologie Unix, on appelle traditionnellement *mailer* un UA).
  Les agents de transport des messages typiques sous Unix sont `sendmail` (une partie de `sendmail` est dédiée au protocole SMTP), `uucp` pour l'acheminement vers un site UUCP, ou encore `/bin/mail` pour la remise physique.

– une boîte aux lettres.
  Sur les systèmes Unix, une boîte aux lettres est simplement un fichier dans `/var/mail/` (systèmes d'origine Berkeley) ou dans `/usr/mail/` (systèmes d'origine System V). Dans un tel fichier, tout nouveau message débute par une ligne `From` (sans caractère « `:` »). Cette ligne, qu'on appelle souvent le « `From` Unix » sert à délimiter les différents messages.

Le monde de la messagerie `sendmail` rend assez difficile une perception nette de chacun de ces composants. En particulier, une confusion peut naître à cause des deux points suivants :

– le programme `sendmail` lui-même est un agent de routage plus un agent de transport spécialisé pour le protocole SMTP. Si cette concentration dans un seul programme n'est pas intellectuellement satisfaisante, elle est dictée par un souci d'efficacité.
– le programme `/bin/mail` est utilisé à la fois comme UA (on peut l'utiliser, bien qu'il soit très frustre) et comme agent de transport de messages (puisque c'est lui qui réalise la remise physique). Ce rôle dual est une erreur de conception, et les nouvelles versions de `sendmail` fournissent un programme distinct pour la remise physique (`/usr/libexec/mail.local` sur les systèmes Berkeley 4.4). Certains sites utilisent également le programme `procmail`.

La terminologie X400 utilise le terme *agent de transfert de messages* (MTA) pour une notion qui regroupe, dans notre terminologie, les agents de routage et de transport.

1.2 Format des adresses

Le courrier électronique suppose l'utilisation d'adresses. Une adresse est la désignation d'une boîte aux lettres dans l'Internet ou ailleurs. La première étape dans la compréhension et la configuration d'un agent de routage de courrier est la maîtrise des différents formats d'adresses possibles.

1.2.1 Adresses Internet

Les adresses utilisées dans l'Internet étaient initialement définies par la RFC 822, puis amendées par la RFC 1123. Par la suite, la RFC 2822 a rajeuni ces spécifications. Il n'y a pas de distinction entre minuscules et majuscules (sauf dans les commentaires ou les chaînes de caractères).

– adresses globales
  Ces adresses sont qualifiées de *globales* car elles spécifient un site sans spécifier de chemin pour y arriver. En cela, elles sont valides à n'importe quel point de l'Internet.
  – `jean @ jussieu.fr`
    La première adresse et la plus simple. Il peut y avoir des espaces entre les différents constituants de l'adresse.
  – `jean (Jean Breille) @ jussieu.fr`
    La même adresse, mais avec un commentaire (entre parenthèses) insérée à n'importe quel point dans l'adresse. Le commentaire est ignoré par l'agent de routage pour la prise de décision, mais il doit être laissé dans l'adresse.
  – `"Jean Breille"@jussieu.fr`
    La partie locale de l'adresse (entre guillemets) est considérée comme un seul mot. Le courrier est donc adressé à l'utilisateur « `Jean Breille` » dans le domaine `jussieu.fr`.
  – `<jean@jussieu.fr>`
    Au niveau syntaxique, cette forme correspond à un cas « dégénéré » des adresses avec routage explicite décrites dans le point suivant. Mais la signification de ces adresses est analogue aux adresses globales.

    La présence des caractères « < » et « > » indique, de manière générale, une adresse facilement exploitable par un programme.
  - `Jean Breille <jean@jussieu.fr>`
    Dans ce cas, un commentaire (sans parenthèses) est ajouté à l'adresse. Un programme privilégie ce qui est entre les caractères « < » et « > » et ignore tout le reste.
  - `"Jean Breille" <jean@jussieu.fr>`
    Cette fois-ci, le commentaire est, au niveau syntaxique, un mot unique puisqu'il est placé entre guillemets.

  Le commentaire (entre parenthèses) n'est pas propre aux adresses globales, il peut être inséré dans tout type d'adresse Internet.
- adresses avec routage explicite (*source-routes*)
  - `<@ibp.fr,@uvsq.fr:jean@jussieu.fr>`
    Cette adresse spécifie que le courrier doit être envoyé à `ibp.fr`, qui doit faire suivre à `uvsq.fr`, qui l'envoie finalement à `jean@jussieu.fr`.

  Les adresses avec routage explicite sont fortement déconseillées. Le problème est que beaucoup de sites ne les reconnaissent pas correctement et, par conséquent, elles ne peuvent pas être utilisées en confiance. La RFC 1123, sans les interdire, déconseille formellement leur utilisation : on *ne devrait pas* les utiliser.
  Il faut noter que, du fait des *spams* (voir 1.10, page 34), de moins en moins de sites acceptent ce type d'adresses.
- adresses littérales
  - `jean @ [134.157.0.129]`
    La présence des crochets indique une adresse numérique, qui doit être prise telle quelle. Le courrier doit donc être envoyé à la machine d'adresse indiquée sans autre forme de traitement (en particulier, sans tenir compte des MX, voir section 1.6, page 16).

  Les adresses littérales sont déconseillées.
- extension RFC 1123 : routage avec `%`
  - `jean % jussieu.fr % uvsq.fr @ ibp.fr`

  Le *%-hack* est spécifié dans la RFC 1123 comme une alternative à utiliser préférentiellement aux adresses avec routage explicite.
  Là encore, du fait des *spams*, ces adresses sont devenues obsolètes. Les RFC 2821 et 2822 ne les ont pas reprises.
- extension RFC 1123 : adresse vide
  - `<>`
    Cette adresse, que peu de sites savent encore gérer correctement, est une adresse à laquelle on ne peut pas répondre.

  Lorsqu'un message d'erreur est généré, l'adresse de l'expéditeur du message d'erreur doit être initialisée à cette adresse. La plupart des sites utilisent aujourd'hui des noms comme Postmaster, Mailer-Daemon, etc. Ces adresses peuvent provoquer des boucles de courrier.
- extension des adresses locales (définie par l'usage, pas par une RFC) :
  - `jean+toto@jussieu.fr`
    Il s'agit d'une astuce : le courrier est envoyé à l'utilisateur `jean` (adresse figurant dans l'enveloppe, voir 1.4, page 13) mais l'adresse figurant dans l'en-tête du message est `jean+toto`.

  Cette astuce permet de trier automatiquement le courrier entrant, par exemple avec le programme `procmail`, suivant l'adresse fournie aux correspondants (`jean+toto@jussieu.fr` ou `jean+titi@jussieu.fr`)

La RFC 2822 précise que tout site doit reconnaître l'adresse spéciale `Postmaster`. Cette adresse ne doit pas être seulement reconnue, mais les courriers qui lui sont destinés doivent également être lus de manière régulière. De plus, de nouvelles adresses conventionnelles se répandent actuellement. Parmi elles, par exemple, l'adresse `abuse` destinée à recueillir les plaintes en cas d'usage abusif de la messagerie. La RFC 2142 recense ces adresses.

1.2.2 Autres adresses

Il y a d'autres réseaux que le réseau Internet. Il y a donc d'autres formats d'adresses. La liste ci-dessous ne prétend pas être exhaustive, mais représente quelques cas que l'on est amené à rencontrer de temps à autre. Pour une liste complète, voir [1].

- adresses UUCP
  Les adresses UUCP sont de la forme `site1!site2!...!siten!user`, spécifiant que le courrier doit être envoyé par UUCP à `site1`, qui doit le faire suivre à `site2` et ainsi de suite jusqu'à `siten` qui le remet à l'utilisateur spécifié.
  Ces adresses sont de temps en temps utilisées par des programmes (le démon d'impression, sur SunOS, envoie par exemple des messages d'erreur dans ce format). On ne peut donc les ignorer totalement.
  Il faut mentionner le *uucp project* qui a pour but, depuis de nombreuses années, de remplacer l'adressage UUCP avec routage explicite par un adressage global du type de l'adressage Internet. On aurait ainsi des adresses du type `user@site.uucp`. Toutefois, la méthode préconisée par l'Internet est l'enregistrement des sites UUCP dans le DNS et routage par le biais des MX, ces sites se situant alors à la *bordure* de l'Internet.
  On peut être amené à rencontrer UUCP dans deux occasions :
  - lorsqu'il faut traiter une adresse dans le monde UUCP. Dans le passé, le traitement de ce cas consistait à transmettre le message à un des sites qui ont une connaissance de l'ensemble des sites UUCP et qui acceptait de servir de passerelle ; en France, il n'existe plus à notre connaissance de passerelle accessible librement. Confronté à ce cas, l'utilisateur devra donc fournir l'adresse d'une passerelle explicitement en utilisant la syntaxe définie ci-après.
  - si on a un ou plusieurs sites isolés directement raccordés par UUCP. On est dans le cas d'un site à la bordure de l'Internet. L'adresse est une adresse Internet classique, mais elle doit conduire à un agent de transport de messages utilisant UUCP.

  Une adresse à la syntaxe UUCP peut être également mélangée avec une adresse Internet. On peut obtenir par exemple `site2!...!siten!user@site1`, ce qui signifie que le message doit être envoyé à `site1` qui doit le faire suivre.
- adresses BITNET
  Le réseau BITNET (Because It's Time to NETwork), ou son alter ego EARN (Eureopean Academic Research Network) en Europe utilise des adresses au format : `user@site`, où `site` est un nom de machine comme `frunip62`. Ces adresses sont de moins en moins répandues au fur et à mesure que ces réseaux perdent en importance. En France, où il reste moins d'une dizaine de machines connectées, à ce jour, il subsiste pendant une durée encore indéterminée une passerelle maintenue par le CNUSC et le CRU.
  Dans l'Internet, la méthode la plus répandue pour référencer un site BITNET est de suffixer l'adresse par `.bitnet` comme par exemple pour `user@frmop11.bitnet`.
  Le routage est effectué par une passerelle, accessible sur l'Internet, et qui connaît l'ensemble du monde BITNET.
- adresses DECnet
  Il existe plusieurs réseaux basés sur le protocole propriétaire DECnet, comme SPAN ou HEPNET. Les adresses sur ces réseaux sont de la forme `user::machine`. Une tendance actuelle consiste à enregistrer les machines de ces réseaux dans le DNS comme pour UUCP.
  Dans le cas contraire, étant donné qu'il n'y a pas *un* réseau, mais plusieurs, on ne peut faire comme pour BITNET ou UUCP, c'est-à-dire tout confier à une passerelle unique implicite. Au contraire, il faut recourir à une adresse de la forme `user::machine@passerelle.dans.internet`.
- adresses X400
  Les adresses X400 sont une suite de couples *champ=valeur* séparés par des caractères « `/` ». Là encore, lorsqu'un site X400 est enregistré dans le DNS, aucune adresse spécifique n'est à gérer.

La disparité des adresses est fâcheuse, et se traduit par autant de traitements au cas par cas dans les agents de routage de courrier. Toutefois, on peut constater que le succès fulgurant de l'Internet et de la communication électronique amène une harmonisation du format des adresses autour des adresses de type RFC 2822.

1.3 Format des messages

Un message est constitué de deux parties distinctes :

- un *en-tête*
  L'en-tête est une suite de champs, définis par la RFC 2822.
- un *corps*
  Le corps est le contenu du message proprement dit. Il est séparé de l'en-tête par au moins une ligne vide. Pendant longtemps, le contenu du corps n'était réglementé que par la RFC 2821 (lignes de 1000 caractères au maximum et caractères tronqués à 7 bits). Le standard MIME (voir section 1.8, page 25) définit dorénavant une structure pour le corps.

Le reste de cette section décrit le format des en-têtes.

Chaque champ d'en-tête est constitué d'un mot-clef, du caractère « `:` », et des informations du champ. Si un champ est trop long, il peut continuer sur la ou les ligne(s) suivante(s). Dans ce cas, chaque ligne de continuation doit commencer par un espace ou une tabulation.

Les caractères suivants ont une signification particulière dans les champs :

Caractère	**Signification**
`\`	le caractère qui suit est spécial
`"` ... `"`	pas d'interprétation à l'intérieur
`(` ... `)`	commentaires

Les champs d'en-tête standardisés[1] sont :

- `Return-Path`
  Ajouté lors de la remise physique du message, c'est-à-dire lors du dépôt dans la boîte aux lettres finale par le dernier agent de transport, pour identifier le routage vers l'expéditeur ; l'adresse qui est indiquée est celle de l'expéditeur dans l'enveloppe (voir section 1.4) ;
- `Received`
  Ajouté par chaque agent de routage le long du chemin emprunté par le message pour signer et tracer ce chemin en cas de problème. Chaque agent peut apporter les informations suivantes :

`from`	site émetteur
`by`	site récepteur
`via`	chemin physique
`with`	protocole utilisé
`id`	identification du message pour le récepteur
`for`	forme initiale

  Par exemple :

[1] La RFC 2076 recense la quasi-totalité des champs d'en-tête utilisés dans l'Internet, normalisés ou non.

```
Received: from shiva.jussieu.fr by soleil.uvsq.fr
        (8.12.1/jtpda-5.4) with ESMTP id JAA28176
        for <pda@prism.uvsq.fr>; Fri,  23 Nov 2001 09:30:12 +0100
```

- `From`
  Identité de l'expéditeur (la personne qui a souhaité que le message soit envoyé), placée par l'UA de l'émetteur. Attention à la confusion possible entre ce champ (avec caractère « `:` ») et la chaîne `From` placée dans les boîtes aux lettres par le programme de remise physique `/bin/mail` sur les systèmes Unix : ces deux chaînes sont proches, mais ont un sens différent (le premier est un champ d'en-tête, véhiculé avec le message, le deuxième est un délimiteur dans la boîte aux lettres) ;
- `Sender`
  Identité de l'expéditeur réel : la personne (personne physique ou processus) qui a composé et envoyé le message, et qui reçoit les messages d'erreur liés au routage du message. Ce champ ne doit pas être utilisé dans les réponses. Le champ `Sender` n'est utile que s'il diffère de `From`. Si ce n'est pas le cas, le champ `Sender` est formellement déconseillé ;
- `Reply-To`
  Adresse de retour, placée par l'expéditeur, utilisée par le destinataire pour les réponses. Si ce champ n'est pas spécifié, `From` est pris par défaut ;
- `Date`
  Date d'expédition, placée par l'UA de l'expéditeur ;
  - jour de la semaine optionnel
  - quantième (2 chiffres)
  - mois (3 lettres)
  - année (4 chiffres)
  - heure
  - fuseau horaire

  Exemple : `Date: Fri, 23 Nov 2001 09:30:15 +0100`
- `To`
  Destinataires principaux du message, spécifiés par l'expéditeur à l'aide de son UA ;
- `Cc` (*carbon copy*)
  Destinataires auxiliaires du message, spécifiés par l'expéditeur à l'aide de son UA ;
- `Bcc` (*blind carbon copy*)
  Destinataires auxiliaires, spécifiés par l'expéditeur à l'aide de son UA. Ce champ n'est pas transmis aux destinataires spécifiés par `To` et `Cc` ;
- `Message-Id`
  Identificateur unique du message, placé par le premier agent de routage, servant à référencer le message. Cet identificateur est souvent constitué d'une partie unique sur l'Internet (par exemple un nom de machine) et d'une partie unique sur la machine (par exemple une date et un identificateur de processus) ;
- `In-Reply-To`
  En cas de réponse, référence au message original placée automatiquement par l'UA de l'expéditeur de la réponse ;
- `References`
  Identification de messages (identificateurs uniques) précédemment envoyés et cités en référence ;
- `Keywords`
  Mots-clefs séparés par des virgules, spécifiés par l'expéditeur à l'aide de son UA ;
- `Subject`
  Sujet du message, spécifié par l'expéditeur à l'aide de son UA ;
- `Comments`
  Commentaires (aucune utilité pour le routage du message), spécifiés par l'expéditeur à l'aide de son UA ;
- `Encrypted`
  Indique que le message est chiffré et spécifie la méthode utilisée ;
- `X-`???
  Les champs commençant par `X-` ne sont pas définis et sont réservés pour les extensions non encore standardisées

ou pour des champs laissés aux utilisateurs.

Lorsque les messages sont re-routés par un utilisateur si son UA le lui permet, certains champs peuvent être ajoutés. Ce sont :

`Resent-From`

`Resent-Sender`

`Resent-Reply-To`

`Resent-To`

`Resent-Cc`

`Resent-Date`

`Resent-Bcc`

`Resent-Message-Id`

Le standard MIME définit également des champs d'en-tête. Ceux-ci sont décrits plus loin (section 1.8).

En outre, certains champs sont définis par `sendmail`. Ils ne sont pas conformes à la RFC 2822, et sont généralement en voie d'obsolescence, mais peuvent encore être rencontrés dans des messages. Il s'agit de :

– `Apparently-To`
  Si aucun champ d'en-tête ne spécifie de destinataire (ce qui est contraire à la RFC 2822), `sendmail` déduit le destinataire de l'enveloppe (c'est-à-dire le nom passé en paramètre à `sendmail`, ou le nom qu'il reçoit par le protocole SMTP).
– `Return-Receipt-To`
  Ce champ n'est pas ajouté par `sendmail`, mais par l'expéditeur du message. Lorsque `sendmail` réalise la remise physique du message et que ce champ est présent, il envoie un message de confirmation de remise (ce qui ne doit pas être confondu avec un accusé de réception) à l'adresse indiquée. L'utilisation de ce champ est fortement déconseillée pour deux raisons essentielles : d'une part, il n'est reconnu que par `sendmail`, c'est-à-dire par une seule implémentation, et d'autre part, il peut générer des boucles de courriers lorsqu'une liste de diffusion est en jeu.
  Ce mécanisme (et donc ce champ) est rendu obsolète par le DSN (*Delivery Status Notification*, voir section 1.7.3, page 22) décrit dans les RFC 1891 à 1894, et implémenté par la version 8.8 de `sendmail`.
– `Full-Name`
  Ce champ contient le nom complet de l'expéditeur.
– `Errors-To`
  Si ce champ est présent, il indique la destination des messages d'erreur. Les versions récentes de `sendmail` rendent ce champ obsolète.
– `Precedence`
  Ce champ donne la priorité devant être affectée au traitement du message. C'est normalement un mot-clef défini dans `sendmail.cf`. Par exemple : `Precedence: list` indique que le message a été transmis par une liste de diffusion, et doit donc être traité avec une basse priorité.

1.4 Enveloppe

Le message comprend l'en-tête, dans lequel se trouvent des informations utiles. Mais le problème est que ces informations ne sont d'aucune utilité pour déterminer le destinataire ou l'expéditeur (pour renvoyer un message d'erreur par exemple). En effet, si le champ To, par exemple, était utilisé pour acheminer le message jusqu'à sa destination, cela poserait de gros problèmes. Considérons un exemple :

```
From: paul@uvsq.fr
To: jean@jussieu.fr, jacques@urec.fr

bla bla bla
```

Ce message est envoyé à partir du site `uvsq.fr`. Si le champ `To` est utilisé, le message est envoyé à `jussieu.fr` et à `urec.fr`. Lorsque le message arrive à `jussieu.fr`, si le champ `To` est utilisé, le message est remis à `jean`, mais est également transmis à `urec.fr` puisque le champ `To` contient deux destinataires. Le site `urec.fr` reçoit donc maintenant deux exemplaires de ce courrier. S'il utilise, lui aussi, le champ `To`, il va envoyer une copie de ces deux courriers à `jean@jussieu.fr`. On voit donc que le système s'emballe, et qu'une boucle de courrier est créée.

Il convient donc de véhiculer une nouvelle information. Cette information, par analogie avec la poste, s'appelle l'*enveloppe*. Comme pour le facteur, qui n'ouvre pas une lettre pour la remettre à son destinataire, l'enveloppe accompagne le message pour le router.

Cette enveloppe est immatérielle : elle n'est pas stockée dans le message lorsqu'il est dans la boîte aux lettres, aussi les utilisateurs n'en ont pas conscience. C'est en fait :

- un argument de l'agent de routage de messages qui indique le ou les destinaitaire(s) du message, ainsi que l'identité du processus qui indique l'expéditeur ;
- des informations qu'un agent de transport de messages transmet lors d'un échange avec un autre agent de transport (par exemple lorsque deux `sendmail` communiquent par SMTP) ;
- une information utilisée par le programme de remise physique pour placer le message dans la boîte aux lettres d'un utilisateur.

1.5 Protocole SMTP

Le protocole SMTP (*Simple Mail Transfer Protocol*) était initialement défini dans la RFC 821, puis amendé par certaines autres RFC donc la RFC 1123. La RFC 2821 a rajeuni les spécifications. Bien qu'en théorie, il ne soit pas nécessaire de connaître le protocole, il peut de temps à autre s'avérer utile d'analyser des traces ou d'interagir directement avec une implémentation de SMTP.

1.5.1 Exemple

Avant de détailler le protocole lui-même, examinons un exemple. Les commandes spécifiées par le site émetteur sont en gras, les réponses du site récepteur sont en caractères normaux.

220 shiva.jussieu.fr Sendmail 8.12.1/jtpda-5.4 ready at Fri, 23 Nov 2001 09:30:15 GMT
HELO soleil.uvsq.fr
250 shiva.jussieu.fr Hello soleil.uvsq.fr, pleased to meet you
MAIL FROM: <paul@uvsq.fr>
250 <paul@uvsq.fr>... Sender ok

RCPT TO: <jean@jussieu.fr>
250 <jean@jussieu.fr>... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
From: paul@uvsq.fr (Paul Ochon)
Subject: essai
To: jean@jussieu.fr (Jean Breille), "Jacques Selere" <jacques @ urec.fr>
Date: Fri, 23 Nov 2001 09:30:05 +0100

ceci est un essai
.
250 Mail accepted
QUIT

La première ligne (`220 shiva.jussieu.fr`...) est envoyée par le récepteur lorsque la communication est établie. Le programme qui implémente le protocole (ici `sendmail`) s'annonce. Il envoie un code numérique (220 signifie qu'il est disponible) qui lui seul a une signification. La suite, pour information seulement, indique le nom de la machine, le programme (`sendmail`), le numéro de version du programme (8.12.1) et de sa configuration (jtpda-5.4), et la date.

La deuxième ligne (`HELO soleil.uvsq.fr`) est transmise par l'émetteur. Il s'identifie en donnant le mot-clef `HELO` (avec un seul L) suivi de son adresse. La troisième ligne est envoyée par le receveur. Elle consiste en un code numérique (250 indique que l'action demandée est effectuée correctement) suivi de l'adresse du receveur. La suite est pour information seulement, elle est ignorée par les deux programmes.

Les quatre lignes suivantes (`MAIL FROM` jusqu'à `250 <jean@jussieu.fr>`) spécifient l'*enveloppe*.

La ligne suivante (`DATA`) indique au receveur que l'émetteur envoie le message proprement dit. Ce message est, comme nous l'avons vu, constitué de l'en-tête et du corps. Le message, après la réponse positive (code `354`) du receveur, est envoyé par l'émetteur, suivi d'un point seul sur sa ligne. Le receveur répond ensuite par le code `250` signifiant que l'opération a réussi.

La dernière ligne est envoyée par l'émetteur. C'est le mot-clef `QUIT` qui précède la déconnexion.

1.5.2 Commandes

Les mots-clefs (ou commandes) du protocole SMTP sont listés ci-dessous. Chaque mot-clef est cité avec son nom, sa syntaxe (entre parenthèses, sans distinction entre minuscules et majuscules) et sa signification.

- Hello (`HELO` *site-émetteur*)
  Commence une session SMTP par une identification des deux parties. Le récepteur s'identifie dans sa réponse ;
- Mail (`MAIL FROM:` *expéditeur*)
  Commence une nouvelle transaction. Spécifie l'expéditeur, pour le renvoi des messages d'erreur éventuels ;
- Recipient (`RCPT TO:` *destinataire*)
  Spécifie un destinataire. Cette procédure peut être répétée autant de fois que nécessaire. Si le destinataire n'est pas valide, un code d'erreur est renvoyé aussitôt ;
- Data (`DATA`)
  Envoie le message (en-tête + corps) terminé par une ligne ne contenant qu'un point.
  Pour ne pas interférer avec le message à transmettre, toute ligne du message contenant un point en première position est modifiée en insérant un point supplémentaire en début. Ce point supplémentaire est supprimé par le

récepteur (*hidden dot algorithm*) ;
- Send (`SEND FROM:` *expéditeur*)
  Commence une nouvelle transaction (de manière analogue à `MAIL`) pour envoyer un message sur un terminal ;
  Note : non supporté par `sendmail`.
- Send or mail (`SOML FROM:` *expéditeur*)
  Commence une nouvelle transaction (de manière analogue à `MAIL`) pour envoyer un message sur un terminal ou dans une boîte aux lettres si le destinataire n'est pas connecté ;
  Note : non supporté par `sendmail`.
- Send and mail (`SAML FROM:` *expéditeur*)
  Commence une nouvelle transaction (de manière analogue à `MAIL`) pour envoyer un message sur un terminal et dans une boîte aux lettres ;
  Note : non supporté par `sendmail`.
- Reset (`RSET`)
  La transaction courante est annulée, l'ensemble du logiciel est réinitialisé, un nouveau message peut donc être envoyé sur le même canal ;
- Verify (`VRFY` *destinataire*)
  Demande au site récepteur de vérifier que le destinataire est une adresse valide ;
- Expand (`EXPN` *destinataire*)
  Demande au site récepteur, d'une part de vérifier que le destinataire est une adresse de liste de diffusion valide, et d'autre part de renvoyer les identités des membres de la liste ;
- Help (`HELP`)
  Help (`HELP` *commande*)
  Renvoie la liste des commandes SMTP admises par le site récepteur ;
- Noop (`NOOP`)
  Ne fait rien ;
- Quit (`QUIT`)
  Termine la session SMTP ;
- Turn (`TURN`)
  Demande au site récepteur de devenir site émetteur. Le site émetteur devient site récepteur ;
  Note : non supporté par `sendmail`.

1.6 Interactions avec le DNS

Le DNS (*domain name system*) est le mécanisme distribué de nommage dans l'Internet. Le protocole SMTP est bien sûr dépendant du DNS, comme tout protocole utilisant des noms de sites. Mais le DNS comprend une caractéristique conçue spécialement pour le courrier électronique : les *resource records* (RR) de type MX (pour *Mail eXchanger*).

Un RR de type MX associé à un site S a pour but d'indiquer aux sites qui voudraient envoyer un courrier à S de ne pas lui envoyer directement, mais de l'envoyer à une autre machine citée dans le MX.

Par exemple :

```
cezanne.prism.uvsq.fr  IN      MX  10  soleil.uvsq.fr
```

Ce MX indique que tout courrier adressé à `cezanne` doit être en réalité envoyé à `soleil`. Toute implémentation de SMTP conforme doit obéir à ce MX.

Dans la pratique, à quoi sert un MX ?

– à router un message pour un site n'ayant pas de connectivité Internet. Par exemple :

```
frmug.fr.net  IN      MX  10  soleil.uvsq.fr
```

a pour effet de demander à tous les sites de l'Internet de router les courriers à envoyer à `frmug` vers `soleil`. Dans cet exemple (obsolète), `frmug` n'était pas sur l'Internet, mais disposait d'une connexion UUCP avec `soleil`, et était donc un site situé à la bordure de l'Internet (voir section 1.2.2).
– à recevoir les courriers adressés à un domaine. Par exemple :

```
uvsq.fr       IN      MX  10  soleil.uvsq.fr
```

Un courrier adressé au domaine `uvsq.fr` est en réalité reçu par `soleil`, ce qui permet d'avoir (si `soleil` est bien configuré) des adresses du genre `user@domaine`.
– à centraliser la distribution du courrier. On peut ainsi fiabiliser la distribution en passant par une machine maintenue, administrée et disponible qui re-route les courriers à l'intérieur d'un campus. La machine qui centralise le courrier doit donc utiliser les MX pour envoyer les courriers vers l'extérieur du campus, et ne doit pas utiliser les MX à l'intérieur pour rerouter les courriers vers les différentes machines (puisque c'est cette machine centralisatrice qui est sensée être le relais).
Par exemple, toutes les machines de l'Université de Versailles/St Quentin en Yvelines ont la définition suivante :

```
cezanne.prism.uvsq.fr IN      MX  10  soleil.uvsq.fr
```

La machine `soleil.uvsq.fr` devant transmettre les courriers destinés à l'intérieur de l'Université, elle ne doit surtout pas utiliser les MX sur le campus. En revanche, lorsqu'il s'agit d'envoyer un message à l'extérieur, elle doit bien sûr les consulter.
Ce rôle centralisateur peut être renforcé, pour accroître la sécurité, par des filtres sur le routeur d'entrée de l'organisation.
– à re-router les courriers en cas de problème. Par exemple :

```
jussieu.fr    IN      MX  10  shiva.jussieu.fr
jussieu.fr    IN      MX  20  soleil.uvsq.fr
```

Les MX sont triés par priorité. Plus le nombre est proche de 0, plus forte est la priorité. Lorsqu'un site Internet essaye d'envoyer un courrier à `jussieu.fr`, il tente d'abord de l'envoyer à `shiva`. Si ce n'est pas possible, par exemple si le campus de Jussieu est coupé du monde extérieur par suite d'un dysfonctionnement du réseau, le site essayera de l'envoyer à `soleil`.
L'intérêt d'une telle manipulation peut paraître réduit : en effet, le protocole SMTP garantit que si un site n'est pas joignable, le message est mis en file d'attente. Mais si, au bout d'une journée, le campus de Jussieu redevient accessible, plusieurs milliers de machines essayeront de se connecter au serveur SMTP de `shiva` sur une très courte durée, ce qui se passera forcément très mal. Un « MX de secours » permet d'alléger cette reprise. Si `shiva` est indisponible, les courriers sont envoyés à `soleil` pendant toute la durée de la coupure. Lorsque `shiva` redevient disponible, elle n'aura à supporter qu'une seule connexion, ce qui est tout à fait raisonnable.
Il y a cependant quelques inconvénients à une telle démarche :
  – les courriers peuvent transiter par un site tiers en cas de panne, ce qui peut amener des problèmes de confidentialité si le site tiers n'est pas sûr ;
  – si l'attente devient longue, la charge du site de secours peut s'alourdir.

Malgré ces inconvénients, le fait d'avoir un site de secours est une bonne pratique. Attention toutefois : une telle opération ne peut se faire qu'avec l'accord de l'administrateur du site concerné.

Dans l'exemple ci-dessus, nous n'avons pas décrit commnt la machine `soleil.uvsq.fr` traite les courriers reçus pour Jussieu. Par défaut, le programme `sendmail` les met en file d'attente, en attendant que `shiva` soit à nouveau disponible. Une pratique plus astucieuse est décrite en 3.6.10 (page 88).

La RFC 974 définit le traitement des MX. Lorsqu'une machine A essaye d'envoyer un courrier à une autre machine B, elle interroge le DNS et récupère en retour une liste de MX.

- si la liste est vide, A ouvre une connexion directe avec B ;
- sinon, la liste est triée suivant les priorités ;
- si A figure dans la liste, toutes les autres machines de même priorité ainsi que les machines moins prioritaires sont retirées de la liste ;
- si la liste est maintenant vide, il s'agit d'une erreur (A est considérée comme étant un routeur pour B, mais A ne sait pas router spécialement les messages pour B) ;
- sinon, A essaye d'ouvrir une connexion directe avec les sites de la liste, en partant du plus prioritaire.

Plusieurs erreurs doivent être évitées lorsqu'on utilise des MX :

- un MX doit pointer sur un nom canonique. En particulier, un MX ne doit pas pointer sur un autre MX, ou sur un CNAME (alias). Le dernier cas est géré correctement par la version 8 de `sendmail`, mais un cas particulier n'est pas le cas général sur l'Internet ;
- une machine agissant comme routeur de courrier pour un domaine doit être capable de gérer les courriers dans son domaine sans tenir compte des MX ;
- il existe un cas spécial, appelé *wildcard MX*. Par exemple :

```
*.frmug.fr.net IN      MX  10  soleil.uvsq.fr
```

  Ce dispositif permet de re-router les messages adressés à `toto.frmug.fr.net` sans que la machine `toto` soit déclarée dans le DNS à l'intérieur du domaine `frmug.fr.net`. Les règles de gestion des *wildcard MX* ne sont pas toujours comprises (en particulier, ces MX ne s'appliquent qu'aux sites qui n'ont pas d'autre RR dans la zone correspondante). La seule règle importante est qu'il **ne faut surtout pas les utiliser**, sauf pour rendre visible au niveau courrier tout un domaine non connecté à l'Internet et non enregistré dans le DNS.

Enfin, une recommandation utile. Qu'on utilise les MX ou non, une bonne pratique est d'avoir un MX sur chaque machine. Par exemple :

```
soleil.uvsq.fr IN      MX  0  soleil.uvsq.fr
```

Le but est de minimiser les requêtes au DNS. Pour comprendre l'intérêt, supposons qu'il n'y ait pas de MX. Une machine voulant envoyer un message à `soleil` procéderait en deux étapes : interrogation du DNS pour obtenir le MX, conduisant à un échec, puis à nouveau interrogation du DNS pour trouver le RR de type A (c'est-à-dire l'adresse IP). Dans ce cas, on voit qu'il y a deux recherches coûteuses.

Si maintenant, on met un MX comme spécifié ci-dessus, la première requête donnerait une réponse, enrichie avec des informations additionnelles telles que le ou les RR de type A correspondant (dans la partie « informations complémentaires » de la réponse). Dans ce cas, il n'y a plus qu'une seule recherche.

1.7 Extended SMTP

1.7.1 Limitations de SMTP

Les implémentations de SMTP conformes à la RFC 821 initiale doivent avoir :

– messages sur 7 bits (tronqués explicitement à 7 bits)
– nom d'utilisateur $<$ 64 caractères
– nom de domaine $<$ 64 caractères
– nombre de destinataires $<$ 100
– lignes $<$ 1000 caractères

Certaines implémentations de SMTP peuvent être moins restrictives. On a vu apparaître, dans les quelques années qui ont précédé la sortie de la version 8 de `sendmail`, des versions qui ne tronquaient pas le huitième bit : ces versions n'étaient clairement pas conformes à la RFC 821.

Le problème est que SMTP était un protocole fermé, sans possibilité de modification, donc d'évolution. Compte-tenu des besoins immédiats, tels que l'échange de courriers contenant des caractères accentués, et des besoins ultérieurs qui ne manqueront pas d'apparaître, deux approches complémentaires ont été adoptées en 1992 :

– définition de MIME (Multipurpose Internet Mail Extensions) (voir 1.8, page 25) pour être capable, entre autres, et si besoin est, d'échanger des messages contenant des accents (donc sur 8 bits) ou des données binaires sur un canal SMTP conforme strictement à la RFC 821 ;
– définition de ESMTP (Extended SMTP), extension au protocole SMTP original (mais formellement intégré depuis la RFC 2821 au protocole SMTP), afin de pouvoir faire évoluer le protocole tout en gardant la compatibilité avec les implémentations actuelles ou à venir.

Le reste de cette section décrit ESMTP.

1.7.2 Principes de ESMTP

L'idée de ESMTP est d'étendre SMTP sans compromettre la compatibilité avec les implémentations antérieures. La RFC 1651 décrit un nouveau mot-clef, `EHLO` (les lettres E et H sont inversées), qui introduit un dialogue ESMTP si les deux parties le reconnaissent.

Si le serveur distant ne le reconnaît pas, le mot-clef est ignoré[2] et le dialogue continue en SMTP classique : le client ESMTP, à la réception du message d'erreur du serveur, envoie le mot-clef `HELO` traditionnel et la conversation continue.

Si le serveur distant reconnaît `EHLO`, il envoie la liste des extensions qu'il supporte, et que le client peut alors utiliser celles qu'il souhaite.

L'exemple suivant illustre le début d'une connexion ESMTP, dans lequel le client (`shiva.jussieu.fr`) s'annonce avec `EHLO`. En retour, le serveur (`soleil.uvsq.fr`) annonce ses extensions :

220 soleil.uvsq.fr ESMTP Sendmail 8.12.1/jtpda-5.4 ready at Fri, 23 Nov 2001 09:30:15 (GMT)

[2]Certaines implémentations violent la RFC 821 et coupent la communication. C'est très fâcheux mais, heureusement, cela arrive peu souvent. Le kit de Jussieu permet de gérer ces problèmes au cas par cas (voir 3.6.10, page 87).

EHLO shiva.jussieu.fr
250-soleil.uvsq.fr Hello shiva.jussieu.fr, pleased to meet you
250-8BITMIME
250-SIZE
250-DSN
250-ONEX
250-ETRN
250-XUSR
250 HELP
QUIT

On notera, en particulier, la présence de l'extension `XUSR`. Cette extension, comme le montre la première lettre (`X`) n'est pas standardisée : elle est propre à `sendmail`.

Certaines extensions peuvent ajouter des commandes au protocole (extension `ETRN` par exemple), d'autres peuvent ajouter des paramètres optionnels au message : dans ce cas, les commandes existantes peuvent être étendues. Par exemple, pour annoncer qu'un message contient des caractères accentués, l'extension `8BITMIME` ajoute un paramètre à la commande `MAIL`.

1.7.3 Quelques extensions

Les paragraphes ci-après montrent quelques exemples d'extensions parmi celles qui sont définies. Elles ne sont pas toutes encore reconnues et supportées par `sendmail`. La liste ne se veut pas exhaustive, puisque l'extensibilité de ESMTP et le rythme de sortie des nouvelles extensions la rendraient rapidement incomplète.

Commandes optionnelles de SMTP

Les premières extensions proposées par la RFC 1869 qui définit ESMTP sont les commandes optionnelles de SMTP. En effet, ESMTP fournit au client un moyen simple d'apprendre si le serveur les supporte. Ces commandes sont `SEND`, `SOML`, `SAML`, `EXPN`, `HELP` et `TURN` (voir 1.5.2, page 15).

Le programme `sendmail` implémente les extensions `HELP` et `EXPN`[3].

Transport des messages contenant des caractères 8 bits

Parmi les extensions, celle décrite dans la RFC 1652 est sans doute la plus importante pour nous francophones. Il s'agit, pour le client, de spécifier qu'un message contient des caractères 8 bits (accentués ou autres).

Si le serveur propose l'extension `8BITMIME`, alors il accepte de prendre en charge un tel message et garantit qu'il arrivera jusqu'à sa destination sans altération. Ainsi, si le serveur n'est qu'un relais et qu'il doit proposer à son tour le message à un autre serveur SMTP qui ne s'annonce pas `8BITMIME`, alors il doit convertir le message selon le format MIME (voir 1.8, page 25) en `quoted-printable` ou en `base64` pour que le destinataire puisse reconstituer le message original. Cette obligation est décrite dans la RFC 1428.

[3]La commande `EXPN` n'est supportée que si l'administrateur ne l'a pas désactivée dans le fichier de configuration.

Pour proposer un message contenant des caractères 8 bits, le client ajoute un paramètre à la commande `MAIL` de SMTP :

`MAIL FROM:` *expéditeur* `BODY=8BITMIME`

Les valeurs possibles pour le paramètre `BODY` sont `8BITMIME` ou `7BIT`.

Le programme `sendmail` supporte et utilise cette extension.

Transport de binaires ou de gros messages

Les deux extensions spécifiées dans la RFC 1830 permettent de transporter de manière efficace les messages binaires ou les gros messages. L'efficacité est obtenue en éliminant l'expansion dûe à la traduction en `base64` (voir 1.8.3, page 29) et la détection de la fin du message (point seul sur une ligne).

L'extension `CHUNKING` remplace la commande `DATA`, pour envoyer le message, par une ou plusieurs commandes `BDAT`. Chaque commande `BDAT` est accompagnée d'une quantité d'octets qui sont transmis sans traitement immédiatement après. La dernière commande `BDAT` d'un message contient le paramètre supplémentaire `LAST`.

`BDAT` *taille* [`LAST`]

L'extension `BINARYMIME` requiert l'extension `CHUNKING` car elle utilise également `BDAT`. Un paramètre supplémentaire est ajouté à la commande `MAIL` :

`MAIL FROM:` *expéditeur* `BODY=BINARYMIME`

Les valeurs `8BITMIME` ou `7BIT` peuvent également être spécifiées. Le message est ensuite transmis avec `BDAT`.

Ces extensions ne sont pas supportées par la version actuelle de `sendmail`.

Émission des messages en attente

L'extension `ETRN`, définie dans la RFC 1985, est utilisée principalement par les sites qui ne sont pas connectés à l'Internet en permanence. Pour leur envoyer du courrier, il faut passer par un relais (via les MX par exemple, voir 1.6 page 16), connecté en permanence et qui conserve les messages sur ses disques en attendant une connexion temporaire du site destinataire. Lorsque celui-ci ouvre une connexion en tant que client, il peut demander le traitement des messages en attente.

Pour cela, il suffit au site destinataire d'ouvrir une connexion ESMTP vers le relais. Le site destinataire utilise la commande `ETRN` suivie par son nom de site (ou `@`nom pour spécifier tous les messages adressés aux sous-domaines éventuels, ou `#`file pour spécifier nommément une file d'attente), ce qui provoque le traitement de la file d'attente par le relais. Celui-ci ouvre alors une nouvelle connexion pour traiter la file d'attente et transmettre tous les messages qui s'y trouvent en attente.

Le programme `sendmail` supporte et utilise cette extension.

Accusé de remise

Les RFC 1891 à 1894 décrivent les DSN (*Delivery Status Notifications*). Il s'agit des *accusés de remise*[4], c'est-à-dire d'un message informant l'émetteur d'un message de son dépôt dans la boîte aux lettres de son destinataire.

L'émetteur peut choisir les conditions d'émission de l'accusé : aucun accusé (même en cas d'échec de la transmission du message), ou alors accusé de réussite, d'échec, ou de retard de remise. L'émetteur peut également choisir entre un accusé contenant, outre un compte-rendu systématique, l'intégralité du message déposé ou seulement son en-tête[5].

Comme pour l'extension `8BITMIME`, si un MTA prend en charge un message demandant un accusé de remise, alors il garantit qu'il y aura émission d'un accusé. En particulier, si le prochain MTA ne supporte pas les accusés de remise, notre MTA doit émettre un accusé d'échec pour prévenir l'expéditeur que l'accusé demandé ne pourra pas lui être remis.

Pour supporter ces accusés, il faut enrichir les informations véhiculées dans l'enveloppe du message original, grâce à des paramètres supplémentaires des commandes `MAIL` et `RCPT`.

> `RCPT TO:` *destinataire* `NOTIFY=`*raison* `ORCPT=`*adresse*

Le paramètre *raison* peut prendre la valeur `NEVER`, auquel cas aucun accusé de remise n'est généré, ou une liste de valeurs parmi `SUCCESS`, `FAILURE` et `DELAY`. Par exemple, si `NOTIFY=SUCCESS,DELAY`, un accusé est remis si le message arrive à bon port, mais également s'il réside longtemps dans la file d'attente d'un MTA sur le chemin.

L'adresse du destinataire peut être réécrite par les MTA intermédiaires, et peut donc ne plus correspondre à ce qu'a tapé l'expéditeur. Afin que l'accusé de remise soit correct, l'adresse originale du destinataire est spécifiée avec le paramètre *adresse*, ainsi que le type d'adresse (les types sont standardisés, et parmi ceux-ci figure le type `rfc822`).

> `MAIL FROM:` *expéditeur* `RET=`*retour* `ENVID=`*identificateur*

Le paramètre *retour* indique la portion du message original que l'accusé doit contenir : les valeurs admises sont `FULL` pour avoir l'intégralité du message, ou `HDRS` pour n'avoir que les en-têtes.

Le paramètre *identificateur*, choisi par exemple par l'UA de l'expéditeur, figurera dans l'accusé. Cela permet à l'UA de retrouver le message original.

L'exemple suivant illustre l'utilisation de ces paramètres. Ici, l'accusé doit être envoyé en cas de succès, et doit spécifier l'adresse `paul@uvsq.fr` comme adresse de destinataire, même si cette adresse est réécrite suite à un alias. L'accusé doit contenir l'intégralité du message.

> 220 soleil.uvsq.fr ESMTP Sendmail 8.12.1/jtpda-5.4 ready at Fri, 23 Nov 2001 09:30:15 (GMT)
> **EHLO shiva.jussieu.fr**
> 250-soleil.uvsq.fr Hello shiva.jussieu.fr, pleased to meet you

[4]Certains autres systèmes de messagerie disposent d'*accusés de présentation*, c'est-à-dire de messages émis par l'UA lorsque celui-ci présente le message au destinataire. Dans l'environnement Unix traditionnel, où une boîte aux lettres est un simple fichier lisible par son propriétaire, il est impossible d'accorder quelque crédit que ce soit à un accusé de présentation. Implémenter un système fiable signifierait rendre la boîte aux lettres inaccessible par son propriétaire, donc une adaptation de tous les UA disponibles actuellement, ce qui n'est pas sans présenter de grandes difficultés.

[5]Le message classique renvoyé par `Mailer-Daemon` en cas d'échec rentre dans ces catégories : il s'agit d'un accusé d'échec, renvoyant l'intégralité du message à son auteur (en plus du compte-rendu).

250-DSN
250 HELP
MAIL FROM: <jean@jussieu.fr> RET=HDRS ENVID=toto
250 <jean@jussieu.fr>... Sender ok
RCPT TO: <paul@uvsq.fr> NOTIFY=SUCCESS ORCPT=rfc822;paul@uvsq.fr
250 <paul@uvsq.fr>... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
From: jean@jussieu.fr (Jean Breille)
To: paul@uvsq.fr (Paul Ochon)
Subject: essai

ceci est un essai
.
250 LAA00602 Message accepted for delivery
QUIT

En supposant que le message arrive à destination, le MTA final enverra le message suivant, dont le format est spécifié par les RFC 1892 et RFC 1894 (voir 1.8.2, page 29) :

```
From: Mailer-Daemon (Mail Delivery Subsystem)
Subject: Return receipt
To: <jean@jussieu.fr>
MIME-Version: 1.0
Content-Type: multipart/report; report-type=delivery-status; boundary="xxxxxx"
Auto-Submitted: auto-generated (return-receipt)

This is a MIME-encapsulated message

--xxxxxx

The original message was received at Fri, 23 Nov 2001 09:30:39 (GMT) from soleil.uvsq.fr

   ----- The following addresses had successful delivery notifications -----
"|/usr/local/bin/procmail -Yf -"  (successfully delivered to mailbox)
    (expanded from: <paul@prism.uvsq.fr>)

   ----- Transcript of session follows -----
"|/usr/local/bin/procmail -Yf -"... Successfully delivered

--xxxxxx
Content-Type: message/delivery-status

Original-Envelope-Id: toto
Reporting-MTA: dns; guillotin.prism.uvsq.fr
Received-From-MTA: dns; soleil.uvsq.fr
Arrival-Date: Fri, 23 Nov 2001 09:30:39 (GMT)

Original-Recipient: rfc822;paul@uvsq.fr
Final-Recipient: rfc822; paul@guillotin.prism.uvsq.fr
Action: delivered (to mailbox)
Status: 2.1.5
Last-Attempt-Date: Fri, 23 Nov 2001 09:30:39 (GMT)

--xxxxxx
Content-Type: text/rfc822-headers
```

```
From: jean@jussieu.fr
Date: Fri, 23 Nov 2001 09:30:04 (GMT)
To: paul@uvsq.fr
Subject: essai

--xxxxxx
```

À partir de la version 8.8, `sendmail` supporte les accusés de remise.

Déclaration de la taille des messages

L'extension `SIZE`, décrite dans la RFC 1870, permet au serveur ESMTP d'annoncer la taille maximum des messages qu'il admet et/ou au client d'annoncer la taille (en octets) du message afin de permettre au site récepteur d'accepter ou de refuser le message.

Le serveur peut annoncer la taille maximum en ajoutant cette valeur derrière l'annonce de l'extension `SIZE`.

Comme pour `8BITMIME`, le client peut annoncer la taille du message en ajoutant un paramètre supplémentaire à la commande `MAIL` de SMTP :

`MAIL FROM:` *expéditeur* `SIZE=`*taille*

La réponse du serveur peut être l'acceptation du message, le refus temporaire, ou le refus définitif.

Le programme `sendmail` n'annonce pas la taille maximum lorsqu'il est serveur, mais annonce la taille des messages lorsqu'il est client.

Commandes à la file

Le protocole SMTP a un défaut mis en évidence dans les liaisons à grande latence, telles que les liaisons transatlantiques : chaque commande attend une réponse en retour. Plus la latence est grande, plus cet aller/retour prend du temps, donc moins le débit est élevé.

L'extension `PIPELINING`, définie dans la RFC 2197, a pour but d'augmenter ce débit. Elle permet au client d'émettre plusieurs commandes sans attendre immédiatement le retour. Ainsi, on peut émettre la commande `MAIL`, autant de commandes `RCPT` qu'il y a de destinataires, et enfin la commande `DATA`. Le serveur renvoie alors les résultats.

Cette extension n'est pas supportée par la version actuelle de `sendmail`.

Reprise de transactions interrompues

Si un échange SMTP est interrompu prématurément, surtout pour de gros transferts, il est intéressant de pouvoir reprendre la transaction au point où elle a été arrêtée. Pour cela, l'extension `CHECKPOINT` définie dans la RFC 1845 ajoute un paramètre supplémentaire à la commande `MAIL` :

`MAIL FROM:` *expéditeur* `TRANSID=`<*identificateur*`@`*client*>

L'identificateur identifie le message sur le site client. Le paramètre `TRANSID` est spécifié lors de la transaction originale. Si cette transaction est interrompue, le client reprend le même identificateur. Le serveur envoie alors l'endroit dans le message où l'interruption s'est produite, et le client peut alors transférer la fin du message.

Cette extension n'est pas supportée par la version actuelle de `sendmail`.

Authentification du client

Pour éviter le relayage par des *spammeurs* (voir 1.10, page 34), une RFC en préparation, basée sur un mécanisme général décrit dans la RFC 2222, a pour but de permettre au client de s'authentifier auprès du serveur. Si le client ne s'authentifie pas, le serveur peut décider de ne pas relayer le message.

L'extension `AUTH` nécessite un nouveau mot-clef `AUTH` à l'initiative du client. Celui-ci précise un mécanisme d'authentification parmi ceux admis par le serveur (annoncés lors de la réponse à `EHLO`). Le serveur répond ensuite avec une ou plusieurs questions (*challenges*) encodées en base 64. Le client doit répondre à ces questions. S'il y répond correctement, l'authentification est réussie. Sinon, toute la suite du dialogue SMTP se déroule comme s'il n'y avait pas eu d'authentification. Lorsque l'échange nécessite une identification du client, celle-ci commence directement avec le mot-clef `AUTH`.

220 soleil.uvsq.fr ESMTP Sendmail 8.12.1/jtpda-5.4 ready at Fri, 23 Nov 2001 09:30:15 (GMT)
EHLO shiva.jussieu.fr
250-soleil.uvsq.fr Hello shiva.jussieu.fr, pleased to meet you
250 AUTH=SKEY
AUTH SKEY c21pdGg=
334 OTUgUWE1ODMwOA==
BsAY3g4gBNo=
235 S/Key authentication successful
...

De plus, la commande `MAIL` admet un paramètre optionnel supplémentaire afin de propager l'identité réelle de l'émetteur, si tant est que tous les serveurs sur le chemin acceptent l'authentification :

MAIL FROM: <jean@jussieu.fr> AUTH=jean@jussieu.fr
250 <jean@jussieu.fr>... Sender ok
...

Cette extension n'est pas supportée par la version actuelle de `sendmail`.

1.8 MIME

Comme nous l'avons vu dans la section 1.7 (voir page 19), la définition originale de SMTP dans la RFC 821 spécifie que le corps du message est constitué de lignes (de longueur au plus égale à 1000 caractères), et que chaque caractère transmis est tronqué à 7 bits.

Cette définition n'autorise évidemment pas la transmission de textes accentués (sauf avec des jeux de caractères ASCII nationaux, obsolètes aujourd'hui). Les très vieilles implémentations de `sendmail` tronquaient effectivement les messages à 7 bits. Quelques implémentations ultérieures ont laissé le huitième bit intact, laissant ainsi passer les caractères accentués en violant la RFC 821. Ces implémentations sont appelées *8 bit transparent* et sont hors de tout standard.

Le standard MIME apporte une réponse, entre autres, au problème des caractères accentués. En réalité, MIME spécifie une nouvelle génération de courrier électronique. MIME répond à une réelle demande puisque certains de ses principes ont été repris dans d'autres contextes (par exemple pour le World-Wide-Web). La définition originale date de juin 1992 (RFC 1341 à 1345) et a été amendée plusieurs fois. En particulier, les RFC 1341 et 1342 sont rendues obsolètes par les RFC 2045 à 2049.

Cette section décrit brièvement le standard MIME et les possibilités nouvelles qu'il apporte. Il ne s'agit pas d'une référence sur le sujet, le lecteur intéressé pouvant se reporter aux RFC décrits ci-dessus.

La RFC 2822 spécifie l'en-tête des courriers, mais pas le corps. Le corps est décrit comme un ensemble de lignes de caractères sur 7 bits, sans plus. L'objet de MIME est de structurer ce corps, en associant un type à chaque information qui s'y trouve ; de plus, comme des informations de plus en plus complexes peuvent être transmises (en particulier des données binaires ou des caractères accentués), MIME spécifie également un système afin de permettre la transmission de ces informations en tenant compte des implémentations les plus anciennes.

Schématiquement, MIME spécifie :

- le type du message, qui peut être du texte, une image, un son, une vidéo, ou encore un agrégat de plusieurs types ;
- le codage du message : plusieurs schémas permettent d'indiquer de quelle manière est codé le message (7 bits, 8 bits, `quoted-printable` ou `base64`).

1.8.1 Exemple

```
From: paul@uvsq.fr
To: jean@jussieu.fr
Subject: essai de MIME
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="toto-tata-titi"

Ceci est un message mime

--toto-tata-titi

Content-Type: text/plain; charset=iso-8859-1
Content-Transfer-Encoding: 8bit

Bonjour Jean, je t'envoie ci-après le texte de la
RFC sur MIME.

--toto-tata-titi

Content-Type: message/external-body; access-type=anon-ftp;
                 site="ftp.jussieu.fr";
                 name="/pub/rfc/rfc/rfc1341.txt"
```

```
--toto-tata-titi
```

Dans cet exemple, on voit clairement les trois nouveaux champs d'en-tête ajoutés pour MIME :

– `MIME-Version`, qui identifie un message au standard MIME, et qui est toujours suivi par 1.0, l'identification du niveau actuel du standard ;
– `Content-Type`, qui spécifie le type, le sous-type et les paramètres optionnels de chaque partie du message ;
– `Content-Transfer-Encoding`, qui spécifie le codage de tout ou partie du message.

Deux champs supplémentaires sont définis, mais ils sont optionnels (non montrés dans cet exemple) :

– `Content-ID`, qui attribue un identificateur unique à chaque partie du message ;
– `Content-Description`, qui permet de spécifier un commentaire attaché à chaque partie.

L'exemple montre un message structuré en deux parties (`multipart/mixed`) séparées par une chaîne unique (spécifiée par `boundary="toto-tata-titi"`). La première partie est un texte classique (`text/plain`) contenant des accents, donc utilisant un jeu de caractère différent de l'ASCII classique (`charset=iso-8859-1`). La deuxième partie est une référence à un objet externe (`message/external-body`) accédé par ftp anonyme (paramètre `access-type=anon-ftp`).

1.8.2 Types, sous-types et paramètres

Chaque type peut comporter un certain nombre de sous-types (dont l'un au moins doit être spécifié). Par exemple, le type `image` est doté des deux sous-types `gif` (pour les images au format GIF) et `jpeg` (pour les images au format JPEG).

Chaque type et sous-type peut avoir des paramètres. Par exemple, un texte (type `text`) sans directive de formattage particulière (sous-type `plain`), doit être accompagné d'un paramètre indiquant le jeu de caractère employé. L'ensemble de ces informations est placé dans le nouveau champ `Content-Type`.

Les types, sous-types et paramètres définis dans la RFC 2046 ne sont pas limitatifs, d'autres peuvent être ajoutés par la suite[6].

Les types définis dans la RFC 2046 sont :

– `text`
  Le type `text` comprend plusieurs sous-types, suivant le degré de formattage du texte :
  – `text/plain` pour les textes non formattés. Il faut noter qu'un courrier à l'ancienne mode (RFC 2822) est implicitement compris comme `text/plain` ;
  – `text/enriched` pour les textes comportant des directives de formattage avec une syntaxe particulièrement simple, similaire à HTML. Par exemple, un courrier contenant :

    ```
    mot en <bold>gras</bold> parmi d'autres
    ```

    sera affiché comme :

    mot en **gras** parmi d'autres

    La définition de ce sous-type est donnée dans la RFC 1563.

  Le seul paramètre de ce type est `charset=` pour spécifier le jeu de caractères utilisé. Parmi les jeux de caractères autorisés, il faut noter que le jeu `us-ascii` ne permet pas l'échange de caractères accentués (l'alphabet

[6] Une addition n'est cependant pas une tâche simple : toute addition provoque implicitement des problèmes d'interopérabilité.

ASCII ne comprend que les 128 premiers caractères), et que seul le jeu `iso-8859-1` permet l'échange de messages composés avec les caractères accentués utilisés en Français[7]. Notons enfin que les jeux de caractères non normalisés comme le jeu PC 8 ne sont pas admis.
D'autres sous-types ont été définis depuis, comme par exemple le sous-type `text/HTML` (voir RFC 2110).
- `image`
  Le type `image` comprend, comme indiqué plus haut, les deux sous-types `image/gif` et `image/jpeg` correspondant aux formats les plus courants de représentation des images.
  Il n'y a pas de paramètre.
- `audio`
  Le type `audio` ne comprend qu'un seul sous-type, `audio/basic`, qui correspond au format de représenation des sons 8 bits, 8 000 Hz, simple canal.
  Il n'y a pas de paramètre.
- `video`
  Le type `video` ne comprend qu'un seul sous-type, `video/mpeg`, correspondant au format de représentation de la vidéo.
  Il n'y a pas de paramètre.
- `message`
  Le type `message` sert à encapsuler d'autres courriers. Les trois sous-types définis sont :
  - `rfc822` pour spécifier l'encapsulation d'un courrier à la syntaxe RFC 2822. C'est typiquement utilisé lorsqu'un routeur de courrier renvoie un message d'erreur : le routeur doit inclure le message original ;
  - `partial` lorsqu'un message MIME est découpé par un UA, un routeur de messages ou un agent de transport lorsque le message est trop gros. Le concept est analogue à la fragmentation des datagrammes IP sur le réseau. Le paramètre `number=` spécifie alors le numéro du fragment de courrier, le paramètre `total=` spécifie le nombre total de fragments et le le paramètre `id=` est un identificateur unique commun à tous les fragments du courrier ;
  - `external-body` pour spécifier une donnée qui se trouve *ailleurs*, c'est-à-dire hors du message. Dans ce cas, il faut indiquer comment faire pour obtenir la donnée : le paramètre `access-type=` indique la méthode (`ftp`, `anon-ftp`, `tftp`, `afs`, `local-file`, `mail-server` et `URL`). D'autres paramètres permettent de préciser cette méthode.
- `application`
  Le type `application` est utilisé pour transmettre des données compréhensibles par des applications annexes. Par exemple, on peut imaginer de transmettre des feuilles de calcul *Excel*, des documents *Word*, etc. À présent, deux sous-types sont définis :
  - `octet-stream` est le plus général : le message contient une suite d'octets qui est censée être une donnée pour une application externe ;
  - `postscript` correspond à un source exécutable par un interprète PostScript.

  Le problème de la sécurité se pose : les courriers sont en effet traités par des programmes. Si on n'y prend pas garde, un utilisateur malicieux peut envoyer un jeu de données malveillant. Il faut donc des contextes d'exécution *sûrs*, ce qui conduit naturellement au concept d'*active mail*.
- `multipart`
  Le dernier type est spécial : il spécifie qu'un message contient en réalité plusieurs sous-messages. Par exemple, un courrier peut contenir un texte, une image et un son. Le message a le type et le sous-type `multipart/mixed`. Un paramètre `boundary=` spécifie une chaîne de caractères dont le but est de séparer chaque sous-partie du message. Chaque sous-partie est composée d'une en-tête miniature spécifiant à nouveau un type et un sous-type MIME.
  Les différents sous-types sont :
  - `multipart/mixed` : les différents constituants sont indépendants et sont présentés par l'UA du destinataire dans l'ordre dans lequel ils sont placés dans le message ;
  - `multipart/alternative` : tous les constituants contiennent la même information, seule la présentation diffère. En fonction de ses capacités, l'UA du destinataire affiche le constituant le plus « joli ». L'exemple typique est un texte envoyé en `text/plain` classique, en `text/enriched` avec des indications de formattage, et en `application/postscript`. L'utilisateur choisit la version la plus lisible pour lui (texte normal s'il lit

[7] Hormis le problème du caractère « œ » et de son équivalent majuscule qui ont été omis (oubliés) par l'ISO.

son courrier sur un Minitel, postscript s'il a un écran X11 avec un *previewer* PostScript, par exemple).
- `multipart/digest` : comparable à `multipart/mixed`, à ceci-près que chaque constituant est lui-même un courrier électronique.
- `multipart/parallel` : les différents constituants sont présentés simultanément à l'utilisateur. C'est typiquement le cas d'une image affichée en même temps que sa légende (un texte) et accompagnée d'un son.
- `multipart/related` : les différents composants constituent un seul et même document, comme par exemple un texte HTML et des icones associées. Ce sous-type est apparu plus récemment (voir RFC 2112).

Une partie d'un message `multipart` peut également être du type `multipart`. On obtient ainsi des messages qui peuvent être relativement complexes. Par exemple, le schéma de la figure 1.1 spécifie que le message est constitué de trois parties, un texte, une alternative et un objet externe. L'alternative est soit une image et un son simultanés, soit une vidéo.

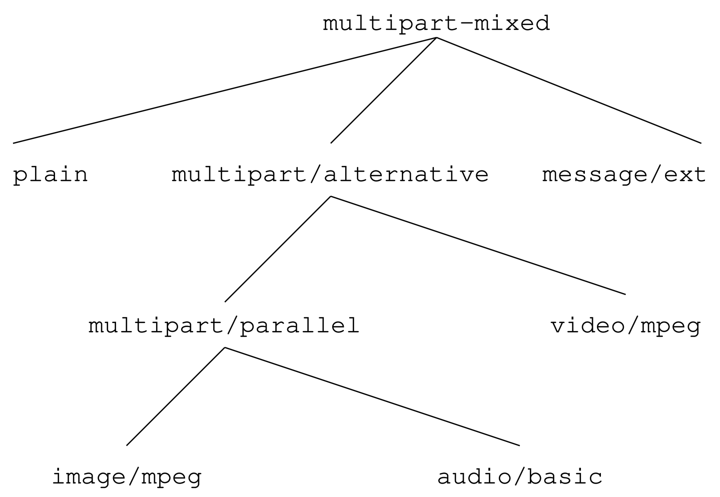

FIG. 1.1 – Exemple de structure de document MIME complexe

D'autres types peuvent être définis ultérieurement. Par exemple, la RFC 1892 définit deux nouveaux sous-types pour supporter les accusés de remise (voir 1.7.3, page 22). Ce sont :

- `multipart/report` : format d'un accusé de remise. Le premier constituant est un message, destiné à l'utilisateur, détaillant la raison de l'émission de l'accusé. Le deuxième constituant reprend les mêmes informations, mais les présente de manière analysable par un programme (par exemple l'UA de l'utilisateur). Le troisième constituant, optionnel, contient le message original (en partie ou en totalité).
- `text/rfc822-headers` : format de la troisième partie de l'accusé de réception.

1.8.3 Codage

Comme nous l'avons vu précédemment, tous les agents de transport de messages ne permettent pas d'envoyer des messages composés de caractères accentués (avec le huitième bit), ou composés de données binaires telles que des sons, des images, des fichiers compressés, etc. car la notion de ligne est toujours présente.

Pour transmettre ces données, il faut les transformer éventuellement dans une forme compatible avec les agents de transport. Cette transformation, si elle a lieu, doit pouvoir être inversée par le destinataire (ou un nœud intermédiaire). Pour cela, on ajoute un champ (`Content-Transfer-Encoding`) à l'en-tête, afin de spécifier le type de codage utilisé pour le corps du message (ou le sous-corps s'il s'agit d'un message multipart).

Ce champ est normalement initialisé par l'UA de l'émetteur, mais il peut également être modifié par un agent de transport intermédiaire (par exemple, un client ESMTP s'apercevant que le serveur ne reconnaît pas l'extension `8BITMIME` doit recoder le message en 7 bits), ou même par l'agent de remise physique lorsqu'il dépose le courrier dans la boîte aux lettres.

À l'heure actuelle, 5 codages possibles sont définis. Ils sont résumés dans le tableau suivant :

Codage	Informations sur	Codée sur	Transportable avec
`7bit`	7 bits	7 bits	SMTP
`quoted-printable`	8 bits	7 bits	SMTP
`base64`	8 bits	7 bits	SMTP
`8bit`	8 bits	8 bits	ESMTP/8bits
`binary`	8 bits	8 bits	ESMTP/binaire

La deuxième colonne indique si l'information d'origine nécessite le huitième bit ou non. La troisième colonne indique si l'information, une fois codée dans le mode spécifié, utilise le huitième bit ou non. La quatrième colonne indique enfin le protocole utilisable pour transférer l'information.

Les deux modes `quoted-printable` et `base64` accomplissent la même fonction : transformer un message contenant des informations sur 8 bits en une suite de caractères 7 bits. Les différences sont la lisibilité et la densité d'information :

- le codage `quoted-printable` consiste, en simplifiant quelque peu, à transformer les rares caractères accentués en une séquence équivalente n'utilisant que des caractères sur 7 bits ; cette transformation est faite en réécrivant un caractère en trois caractères : « = », puis les deux chiffres hexadécimaux du code du caractère ;
- le codage `base64` est comparable au codage `uuencode` traditionnel sous Unix : il consiste à coder 3 octets dans 4, autrement dit découper 24 bits consécutifs (3×8) en 4 paquets de 6 bits, 6 bits suffisant pour représenter un caractère dans un alphabet commun à toutes les machines de la création.

On voit bien que le codage `base64` permet de coder des fichiers binaires avec une surcharge (*overhead*) relativement faible par rapport au codage `quoted-printable` (multiplication de la taille des messages par 4/3 au lieu de de 3 dans le pire des cas). En revanche, un message codé en `base64` est illisible, alors qu'un message codé en `quoted-printable` reste déchiffrable par le destinataire qui n'a pas de décodeur approprié.

Les messages codés en `8bit` nécessitent, pour être véhiculés sur le réseau, un protocole admettant les données sur 8 bits. Seul ESMTP avec l'extension `8BITMIME` (voir page 20) sait le faire correctement.

Le dernier codage, `binary`, est prévu pour transférer des données binaires telles que sons, images, etc. sans modification. S'il est théoriquement transportable avec ESMTP et l'extension `BINARYMIME` (voir page 21), il n'est en pratique pas traité par `sendmail`, et il pose des problèmes de représentation dans les boîtes aux lettres.

1.8.4 Intégration de MIME

Dans un monde parfait, envoyer un message MIME (contenant des accents par exemple) est simple : l'accent reste tel qu'il est, l'UA place dans le paramètre `charset=` le jeu de caractères normalisé (`iso-8859-1` par exemple), le champ `Content-Transfer-Encoding` précise qu'il est codé en `8bit`, et... « roulez jeunesse ! » : l'accent est transmis via ESMTP avec l'extension `8BITMIME` et il arrive dans la boîte aux lettres du destinataire inchangé.

Malheureusement, le monde n'est pas parfait et des implémentations de SMTP peuvent barrer le chemin au huitième bit, faisant ainsi perdre au message toute sa signification.

Concrètement, comment intégrer MIME dans la configuration actuelle de la messagerie électronique sur l'Internet ? Pour répondre à cette question, il faut tenir compte de deux facteurs :

– l'implémentation par le MTA du protocole ESMTP et de l'extension `8BITMIME` (que nous appellerons dorénavant ESMTP8) ou non ;
– le fait que MIME repose à la fois sur l'UA et sur le MTA. Par exemple, l'UA est responsable du typage des messages, mais également du codage initial. De plus, il est préférable (mais pas indispensable) que l'UA connaisse les capacités du MTA utilisé (c'est-à-dire s'il sait recoder des messages au vol ou non) ;

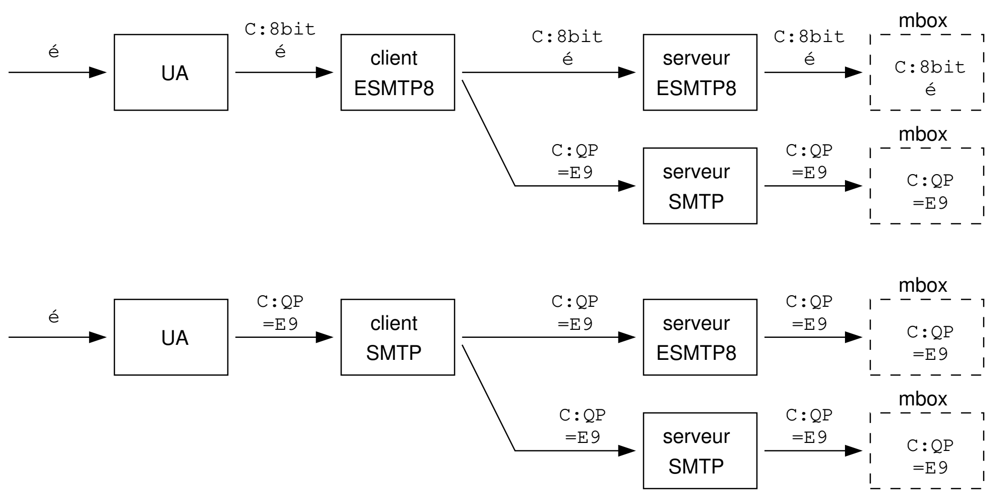

FIG. 1.2 – Migration vers MIME

La RFC 1428 précise les conditions de la migration. Celles-ci sont simplifiées et résumées dans le schéma de la figure 1.2, dans laquelle apparaissent quatre scénarios.

– Premier scénario : l'utilisateur envoie un message contenant un accent (`é`). L'UA, sachant que le MTA est ESMTP8, lui fait confiance et ajoute seulement le champ `Content-Transfer-Encoding` avec le codage `8bit` (représenté sur la figure par « `C:8bit` »). Le MTA (client ESMTP8) dialogue avec le MTA distant (serveur ESMTP8), s'aperçoit qu'il comprend ESMTP8, et lui envoie donc le message sans modification. Le MTA distant dépose alors le message dans la boîte aux lettres du destinataire, sans modification.
– Deuxième scénario : cette fois-ci, le MTA distant ne comprend pas ESMTP8 (soit parce qu'il ne comprend que SMTP, soit parce qu'il comprend ESMTP, mais pas son extension `8BITMIME`). Notre MTA local (toujours client ESMTP8 sur la figure) ne peut donc pas lui faire confiance, ne sachant pas s'il va tronquer le huitième bit (SMTP) ou sachant justement qu'il va tronquer le huitième bit (ESMTP sans `8BITMIME`). Le MTA **recode** donc le message en `quoted-printable` (représenté par `C:QP` sur la figure) et l'accent est donc transformé en `=E9`. Le MTA distant véhicule ensuite cette information inchangée jusque dans la boîte aux lettres du destinataire.
– Troisième et quatrième scénarios : maintenant, l'UA sait que le MTA local (client SMTP sur la figure) n'implémente pas ESMTP8 et ne sait pas recoder les messages au vol. Il ne peut donc pas lui faire confiance (car il ne sait pas si ce MTA ou un autre sur le chemin va tronquer le huitième bit) et doit directement coder l'accent en `quoted-printable` pour donner `=E9`. Le MTA transporte ensuite naïvement le message sans le changer, que

le MTA distant soit ESMTP8 ou non.

À l'heure actuelle, la situation est la suivante :

- les versions de `sendmail` à partir de la version 8.7 gèrent correctement l'extension `8BITMIME` de ESMTP et savent recoder les messages au vol ;
- les versions de `sendmail` à partir de la version 8.8 peuvent éventuellement transformer des corps codés en `quoted-printable` ou `base64` en corps `8bit`, ce qui est intéressant pour les vieux UA qui n'ont pas été adaptés à MIME ;
- bon nombre d'UA récents intègrent le support de MIME (même si leur qualité n'est pas toujours au rendez-vous), mais il reste encore des UA non actualisés, comme le bon vieil utilitaire `mail` par exemple).

Dans ce contexte, quelle démarche adopter ? Il est clair que la solution « légale » consiste à envoyer les messages en `8bits` et laisser `sendmail` transformer éventuellement en `Quoted-Printable` ou en `Base64` (c'est l'option retenue par défaut dans le kit). Pour ne pas pénaliser les utilisateurs disposant de vieux UA, la remise en forme des accents par `sendmail` version 8.8 est un plus appréciable (c'est là encore l'option par défaut dans le kit).

1.9 Lecture déportée

De plus en plus d'utilisateurs lisent leur courrier non pas directement sur Unix avec un UA local, mais depuis un micro-ordinateur (PC ou Mac) avec un UA *déporté*. Cette lecture est rendue possible grâce aux protocoles de lecture à distance : POP3 (*Post Office Protocol* version 3) défini dans la RFC 1939 ou IMAP4 (*Internet Message Access Protocol* version 4) défini dans la RFC 2060.

Dans ce type d'architecture de messagerie, le MTA (par exemple `sendmail` sur Unix) dépose le courrier dans les boîtes aux lettres des utilisateurs. La configuration du MTA n'est pas spécifique à la lecture déportée. Lorsqu'un utilisateur désire lire ses messages, il se connecte au serveur POP3 ou IMAP4 (qui tourne vraisemblablement sur la même machine que `sendmail`), ce qui provoque le lancement d'un programme auxiliaire (le démon implémentant le protocole) pour lire la boîte. Cette configuration ne nécessite que l'ajout d'un démon, sans modifier la configuration du MTA, elle est donc très simple à mettre en œuvre.

Ces protocoles peuvent également être utilisés par des UA sur Unix. On a ainsi le moyen d'éviter le partage des boîtes aux lettres par NFS[8]. Malheureusement, trop peu d'UA disposent de cette possibilité, ce qui en freine l'utilisation, bien que l'on puisse utiliser des programmes comme `fetchmail` (pour un exemple, voir 3.7.4, page 115).

Notre but n'est pas ici de détailler ces deux protocoles, mais de donner une idée de leurs caractéristiques. Pour cela, nous donnons un exemple de dialogue POP3, le plus répandu des deux protocoles[9] :

+OK QPOP (version 2.2) at soleil.uvsq.fr starting. <4145.845637275@soleil.uvsq.fr>
USER pda
+OK Password required for pda.
PASS monmotdepasse
+OK pda has 73 messages (146800 octets).
STAT
+OK 73 146800
LIST 1
+OK 1 998

[8] Le partage de boîte aux lettres par NFS peut conduire à des problèmes allant jusqu'à la perte de messages.
[9] Bien que ceci ne soit plus tout à fait vrai depuis le support de IMAP4 par certains programmes très populaires...

RETR 1
+OK 998 octets
suit le message, avec ses en-têtes, terminé par un point seul sur sa ligne
DELE 1
+OK Message 1 has been deleted.
QUIT
+OK Pop server at soleil.uvsq.fr signing off.

Les commandes du protocole POP3 sont listées ci-dessous. Chaque commande est listée avec son nom, sa syntaxe (entre parenthèses, sans distinction minuscules ou majuscules) et sa signification.

– `USER` *nom*
  Commence une session POP3 par une identification de l'utilisateur.
– `PASS` *mot-de-passe*
  Fournit le mot de passe.
– `APOP` *nom signature*
  L'authentification par mot de passe oblige à faire circuler le mot de passe en clair sur le réseau. Pour éviter cela, le serveur, s'il supporte la commande optionnelle `APOP`, envoie une chaîne différente à chaque connexion lors du message initial (`<4145.845637275@soleil.uvsq.fr>` dans l'exemple). Si l'UA authentifie l'utilisateur par `APOP`, au lieu de `USER`/`PASS`, il ajoute à cette chaîne le mot de passe, en calcule la signature MD5 et l'envoie au serveur. Celui-ci fait la même opération et accepte la connexion si les deux signatures sont égales. Cela évite de faire transiter le mot de passe en clair sur le réseau, et la scène ne peut être « rejouée » puisque la chaîne initiale est différente à chaque connexion.
– `STAT`
  Liste les caractéristiques de la boîte aux lettres (nombre de messages et taille totale de la boîte en octets).
– `LIST`
  `LIST` *numéro*
  Liste les caractéristiques (numéro de message et taille en octets) d'un ou de tous les messages de la boîte aux lettres.
– `RETR` *numéro*
  Affiche un message, terminé par un point seul sur sa ligne.
– `DELE` *numéro*
  Marque un message comme à détruire.
– `LAST`
  Renvoie le numéro du dernier message consulté.
– `RSET`
  Réinitialise le serveur POP3 et annule toutes les actions (y compris la suppression des messages marqués).
– `TOP` *numéro nb-de-lignes*
  Affiche l'en-tête, puis les *nb-de-lignes* lignes du corps du message repéré par son numéro.
– `RPOP` *nom*
  Sous certaines conditions, autorise l'accès au serveur sans passer par les commandes `USER` et `PASS`.
– `UIDL`
  `UIDL` *numéro*
  Cette commande optionnelle sert à demander au serveur d'associer un identificateur permanent à un message, ou à tous les messages d'une boîte aux lettres. L'idée est qu'un client peut alors savoir de manière fiable quels sont les nouveaux messages depuis la précédente connexion.
– `NOOP`
  Ne fait rien.
– `QUIT`
  Termine la session POP3 et supprime les messages marqués pour destruction.

À l'heure actuelle, une des meilleures implémentations de serveur POP3 est celle de la société Qualcomm (qui commercialise également le célèbre UA Eudora pour PC et Mac). Le serveur est disponible sur `ftp.qualcomm.com`.

Le démon `poppassd` pour changer son mot de passe (implémentant un protocole non standardisé, mais très utile pour les utilisateurs d'Eudora) est aussi disponible sur le même serveur.

1.10 Le spam

Un chapitre sur le courrier électronique ne serait pas complet sans une section sur la plaie qui affecte dorénavant une grande partie des utilisateurs : le *spam*. Il s'agit ici essentiellement d'une introduction : le lecteur intéressé est invité à se référer à l'excellent article[10] de Stéphane Bortzmeyer sur ce sujet.

1.10.1 Introduction

Le *spam* est un courrier non sollicité envoyé à de très nombreuses personnes, analogue aux prospectus qui inondent nos boîtes aux lettres (réelles, pas informatiques).

Pour l'anecdode, le terme *spam* a deux origines possibles (et pas forcément exclusives) : pour certains, il s'agit d'une marque américaine d'aliments bon marché de mauvaise qualité, alors que pour d'autres, il viendrait d'un sketch des Monty Python.

Le *spam* est apparu avec l'explosion du nombre d'utilisateurs sur l'Internet. Pour certains nouveaux-venus sur le réseau, Internet représente un vaste espace peuplé de consommateurs prêts à gober leurs publicité et à acheter leurs produits, d'âmes à convertir à une nouvelle secte, d'électeurs prêts à écouter leur propagande, etc. Quand, en plus, la diffusion en masse (*mass-mailing*) coûte 1 000 à 10 000 fois moins cher qu'avec le papier, il est aisé d'imaginer l'effet sur des *marketoïdes* simples d'esprit...

Comment les *spammeurs* obtiennent-ils d'aussi nombreuses adresses ? Toutes les méthodes sont bonnes : analyser les news pour en extraire les champs `From:`, récupérer des sites Web entiers à la recherche d'URL de type `mailto`, lister des adresses avec des moteurs de recherche, etc. Peu de *spammeurs* (ils n'ont généralement pas les compétences suffisantes) obtiennent ces adresses par eux-mêmes, mais nombreux sont les *spams* pour vanter telle liste de *x* millions d'adresses... Le commerce du *spam* est florissant, et certaines sociétés ont pignon sur rue (Cyberpromo est une des plus connues, malheureusement...).

Les amateurs d'acronymes apprécieront la classification des courriers non sollicités en plusieurs catégories :

– UBE (*Unsollicited Bulk Email*) : il s'agit des courriers non sollicités envoyés en masse, c'est-à-dire les *spams*. L'acronyme UBE est un terme générique, quel que soit le sujet (commercial, publicité, chaîne, propagande, etc.) ;
– UCE (*Unsollicited Commercial Email*) : ce sont les courriers purement commerciaux. Tout courrier UCE n'est pas forcément envoyé à de multiples destinataires, donc tout UCE n'est pas forcément un *spam* ;
– MMF (*Make Money Fast*) : les chaînes (interdites en France et aux USA entre autres pays), les recettes miracles pour devenir riche rapidement et sans effort, etc.
– MLM (*Multi-Level Marketing*) : il s'agit d'une variante de la catégorie précédente, à savoir tous les schémas pyramidaux (analogues aux chaînes).

Quelle que soit la catégorie, un *spam* est toujours une nuisance.

Enfin, il faut signaler que le problème du *spam* est apparu en premier dans les *news*. Toutefois, les moyens tech-

[10] http://www.pasteur.fr/other/computer/JRES97/AntiSpam/

niques pour lutter contre le *spam* sont très largement différents dans ce contexte.

1.10.2 Comment lutter contre le spam

La lutte contre le *spam* a plusieurs facettes : pédagogique, boycott, juridique, etc. Nous n'aborderons ici que le strict plan technique.

Reconnaître l'auteur d'un spam

Reconnaître un *spam* est facile. En revanche, en reconnaître l'auteur est nettement plus difficile. De plus il est aisé avec SMTP que se faire passer pour un autre, et l'utilisation de relais (le plus souvent involontaires, voir plus bas) n'arrange pas la lisibilité.

Certaines affaires récentes ont montré que la personne nommée dans le champ `From` n'était pas forcément l'auteur du *spam*. Mais c'est en tous cas celle qui a eu des ennuis, tant informatiques (boîte aux lettres pleine) que personnels (descente du FBI au domicile). Il faut donc se garder des réactions à chaud, et il faut bien maîtriser la lecture des champs `Received` du message.

Adresse « abuse »

L'adresse `Postmaster@site` est obligatoire pour tout site abritant une messagerie SMTP. Une convention tend à se répandre, l'adresse `abuse@site`, utilisée pour signaler les plaintes.

Étant donné que chaque site peut potentiellement devenir la source d'un *spam*, c'est une bonne idée que de reconnaître cette adresse `abuse`. De plus, reconnaître et accepter cette adresse est souvent un gage de bonne volonté : vos correspondants sauront que vous êtes conscient des problèmes que pose le *spam* et seront peut-être plus indulgents avec vous...

Filtrage au niveau personnel

Chaque personne peut développer, à l'aide d'outils comme `procmail` par exemple, un jeu de filtres adapté. L'idée peut être, par exemple, de rejeter tout courrier suspect dans une boîte aux lettres qui n'est pas lue aussi souvent que la boîte aux lettres normale.

Par exemple, l'auteur a mis en place les règles suivantes, radicales mais aussi très efficaces :

```
# reconnaissance prealable des listes de diffusion
# auxquelles je suis abonne

# reconnaissance de cas particuliers
:0:
* ^From.*@.*(mes-amis|mes-fournisseurs).(com|net)
$DEFAULT
```

```
# tout ce qui vient de .net ou de .com est jete
# dans l'antichambre de la poubelle
:0:
* ^From.*@*\.(net|com)
Mail/spam
```

D'autres méthodes sont encore possibles : mécanisme de *liste noire* personnelle pour rejeter les courriers de domaines connus, mécanisme de *score* pour détruire tout ce qui contient tel ou tel mot souvent utilisé par les *spammeurs*, etc.

Filtrage au niveau d'un site

Au niveau d'un site, il n'est pas envisageable d'adopter une solution aussi radicale. En revanche, plusieurs actions permettent d'éliminer une partie des *spams*. Il s'agit de se protéger contre :

– les *spammeurs* bien connus : en établissant une *liste noire* des adresses IP ou des noms de domaines ou d'adresses électroniques de *spammeurs* notoires ; il faut toutefois noter qu'une telle liste est vouée à l'obsolescence rapide, il convient donc de la mettre à jour fréquemment, ou d'utiliser une ou plusieurs listes centralisées (voir RBL, ci-après) ;
– le relayage : beaucoup de sites, par défaut, routent les courriers sans distinguer s'ils le font à bon escient. De plus en plus de *spammeurs* étant dans des listes noires, ils tirent profit de cette naïveté pour envoyer à un domaine innocent leurs millions de courriers à expédier. Le résultat net est que le serveur du site innoncent s'effondre sous la charge, reçoit de nombreuses plaintes et, c'est un comble, se retrouve dans des listes noires un peu partout dans le monde, desquelles il sera très difficile de s'extirper !
– les adresses invalides : beaucoup de *spammeurs*, du fait du coût d'un nom de domaine, fournissent une adresse invalide (comme `machin@become-rich.com` par exemple) ;
– les adresses IP d'expéditeurs non valides : il ne s'agit pas à proprement parler d'une caractéristique des *spammeurs*, mais ils sont cependant nombreux à opérer depuis des machines non enregistrées dans le DNS. Cela peut toutefois empêcher également les communications en provenance de sites honnêtes, mais mal configurés. Une attitude radicale peut géner sur le court terme, mais peut également rendre service à l'Internet sur le long terme, en forçant les sites à bien configurer leur DNS.

En appliquant ces simples règles (avec `sendmail` par exemple, voir 2.5.5, page 48), on n'élimine certes pas tous les *spams* possibles, mais on en filtre une grande majorité.

Les RBL, ou « Realtime Blackhole Lists »

La liste noire de Paul Vixie est apparue en 1997. Elle offrait une arborescence DNS construite à partir des adresses IP de domaines *spammeurs* (ou ayant relayé un *spam*). Ainsi, lorsqu'une connexion venait de la machine d'adresse `a.b.c.d`, il suffisait de regarder si le *resource record* `d.c.b.a.rbl.maps.vix.com` existait dans le DNS. Ainsi, la communauté avait à sa disposition une *liste noire*, maintenue de manière centralisée et diffusée facilement et rapidement. Le kit Jussieu a offert dès 1998 la possibilité d'utiliser cette liste.

En 2000, les règles d'utilisation ont changé : outre la migration du domaine vers `mail-abuse.com`, il faut dorénavant une licence (gratuite dans certains cas, payante dans les autres) rebute nombre de personnes. On a alors vu se développer d'autres listes noires de ce type, avec des règles d'utilisation libres, telles que `relays.ordb.org` ou `inputs.orbz.org`.

Le kit offre la possibilité d'utiliser, très simplement, une ou plusieurs listes noires de type RBL.

Chapitre 2

Sendmail

Ce chapitre décrit `sendmail` sans entrer dans les détails de sa configuration. Seules sont tracées les grandes lignes de ce programme complexe. La dernière section détaille les problèmes liés à la mise au point et aux tests d'une configuration. Pour plus de détails, se référer à la documentation venant avec `sendmail`.

2.1 Introduction

Le programme `sendmail` est une implémentation d'un agent de routage de messages ainsi que d'un agent de transport spécialisé pour le protocole SMTP (voir figure 2.1).

Ce programme a été écrit par Eric Allman, en 1980, alors qu'il travaillait à l'Université de Berkeley. Lorsque Eric a quitté Berkeley, il a arrêté de travailler sur `sendmail`. Le programme est alors resté en l'état (version 5). Plusieurs améliorations ont alors vu le jour indépendamment, la plus connue étant la version IDA. Depuis 1990, Eric a repris son travail sur `sendmail` et a sorti la version 6, qui est très rapidement devenue la version 8 pour des raisons de numérotation de versions internes à Berkeley.

Les principales difficultés liées à la compréhension de `sendmail` sont de plusieurs ordres :

- syntaxe du fichier de configuration assez rudimentaire, ne facilitant pas la lisibilité ;
- connaissance nécessaire des divers formats d'adresses et des protocoles sous-jacents ;
- diversité des configurations de courrier.

Ce chapitre a pour but de présenter le fonctionnement de `sendmail`, sans rentrer dans la syntaxe de son fichier de configuration.

2.2 Composants

Comme le montre la figure 2.1, `sendmail` est en réalité composé de plusieurs éléments :

- un agent de routage de messages

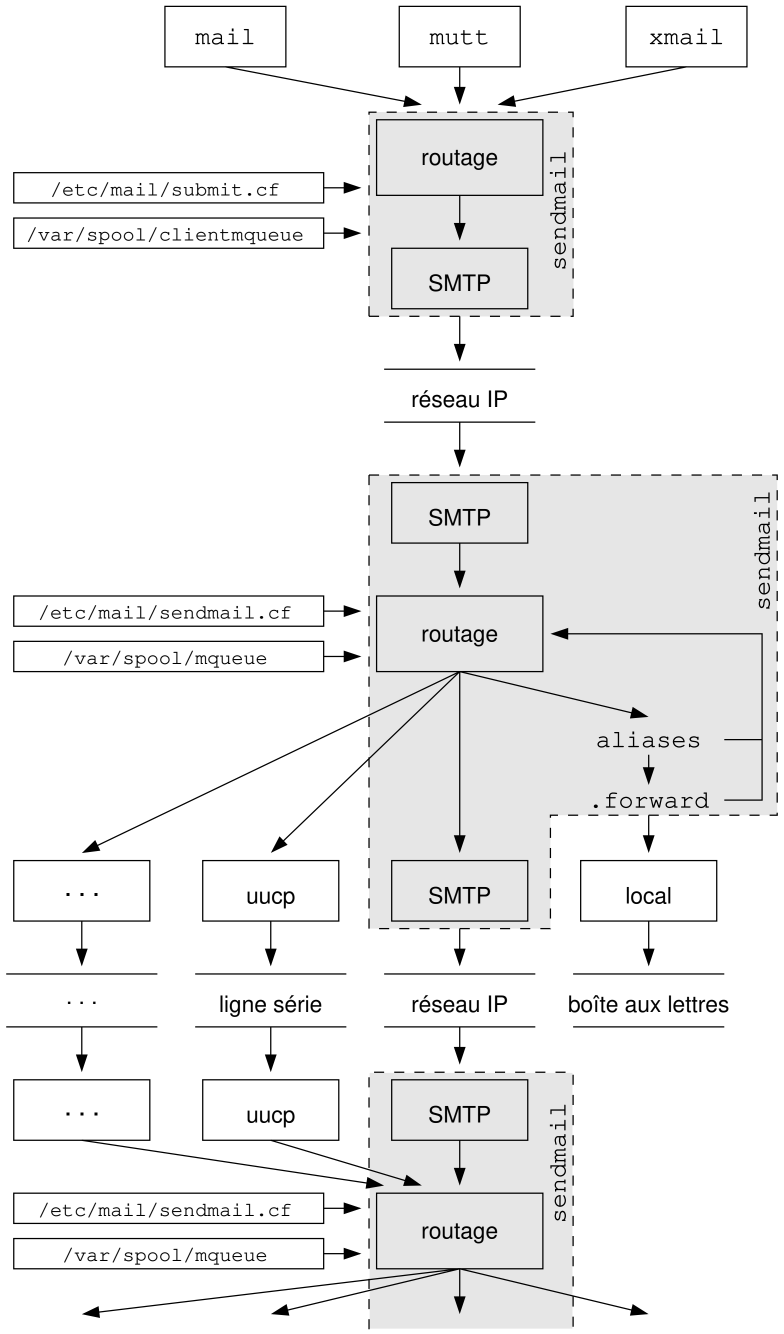

FIG. 2.1 – Situation de `sendmail`

– un client SMTP
– un serveur SMTP
– une partie du traitement des courriers locaux

2.2.1 Fonctionnement en relais de messagerie

Dans son fonctionnement normal de relais de messagerie, `sendmail` attend un message sur le port SMTP, puis le stocke dans la file d'attente `mqueue/` pour ne pas le perdre en cas de problème système. L'agent de routage traite alors ce message, et l'envoie vers le mécanisme de transport adéquat

– si le mécanisme de transport sélectionné est SMTP, `sendmail` assume le rôle de client SMTP pour dialoguer avec la machine distante ;
– si le mécanisme de transport est la remise physique dans une boîte aux lettres locale, `sendmail` effectue quelques vérifications (fichier `aliases`, fichier `.forward` de l'utilisateur) avant d'appeler le programme qui ajoutera le message dans le fichier servant de boîte aux lettres ;
– dans tous les autres cas, `sendmail` se contente d'appeler un programme externe qui se chargera de transmettre le message avec les protocoles appropriés.

2.2.2 Soumission d'un message jusqu'à la version 8.11

La soumission d'un message par un agent utilisateur se faisait par simple appel de `sendmail`, avec comme argument l'adresse du ou des destinataires, ce qui constituait alors une partie de l'enveloppe (voir 1.4, page 13).

Jusqu'à la version 8.11 de `sendmail`, cet appel initié par l'agent utilisateur, donc avec les droits de l'utilisateur, démarrait toute la mécanique de `sendmail`, y compris le stockage du message dans la file d'attente, la lecture des `.forward` des utilisateurs, ou encore l'appel au programme de remise physique qui devait changer d'identité. Pour faire tout ceci, le binaire devait donc avoir le bit *set-user-id* et être propriété de `root`, ce qui était assez dangereux.

2.2.3 Soumission d'un message depuis la version 8.12

Depuis la version 8.12 de `sendmail`, la fonctionnalité de soumission d'un message a été logiquement séparée, bien que figurant toujours dans le même exécutable. La figure 2.2 montre les différentes fonctionnalités de `sendmail`.

Lors d'une soumission de message, `sendmail`, configuré à l'aide d'un fichier de configuration séparé `submit.cf` se contente de se connecter à un serveur SMTP pour envoyer le message. Si le serveur ne répond pas, `sendmail` dépose le message dans une file d'attente (`clientmqueue/`) réservée à cet usage.

Cette instance de `sendmail`, appelée *Message Submission Program*, n'a pas besoin d'avoir les droits de `root` pour fonctionner. Son seul besoin est de pouvoir écrire dans la file d'attente, en cas de besoin. Afin de réduire les failles de sécurité, un groupe spécifique doit être créé (`smmsp`) et l'exécutable `sendmail` peut se contenter d'avoir le bit *set-group-id* et d'appartenir à ce groupe `smmsp`.

Périodiquement, une autre instance de `sendmail`, le *Queue Runner*, surveille cette file d'attente spécifique, et s'il y a des messages en souffrance, tente une connexion vers le serveur SMTP.

Ces deux instances de `sendmail` sont paramétrées avec un fichier de configuration spécifique, `submit.cf`, qui n'a

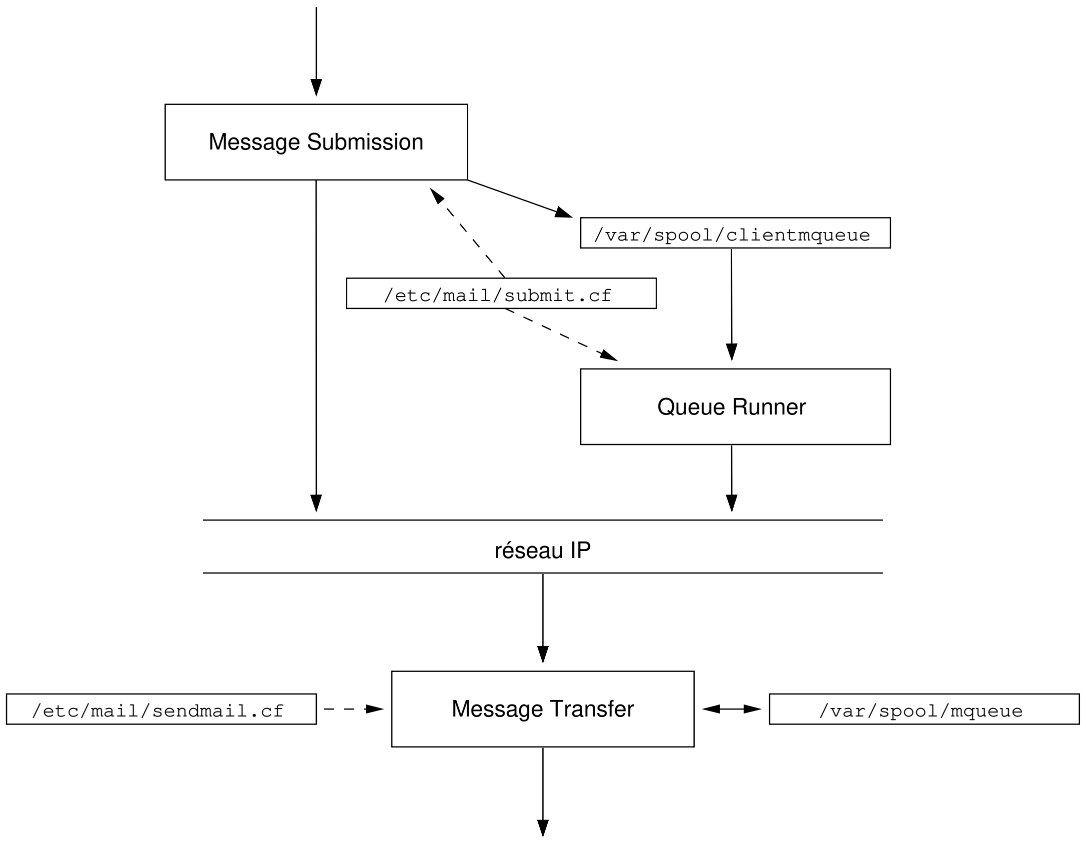

FIG. 2.2 – Soumission d'un message avec les versions de `sendmail` postérieures à 8.12

comme rôle essentiel que de localiser la file d'attente, le serveur SMTP et quelques autres variables.

Enfin, le serveur SMTP constitue le dernier composant : le *Message Transfer Agent*. Il a sa propre file d'attente, `mqueue/`, distincte de la file d'attente précédente, son propre fichier de configuration `sendmail.cf`, et tourne avec les droits de `root` car il est lancé par cet utilisateur au démarrage du système.

2.3 Arborescence

Le programme `sendmail` utilise un certain nombre de fichiers et de répertoires. La plupart des chemins sont modifiables. Nous donnons ici une arborescence type pour une installation standard de la version 8.12 de `sendmail` sur une version issue de Berkeley 4.4.

– répertoire `/etc/mail/` :
  – `sendmail.cf`
    Fichier de configuration. Ce chemin est modifiable sur la ligne de commande de `sendmail` (argument `-C` d'appel), mais cela n'est utilisable qu'à des fins de test : les UA appellent en effet `sendmail` sans donner de paramètre autre que les adresses des destinataires.
  – `sendmail.fc`
    Configuration « compilée », obsolète à présent. Les anciennes versions de `sendmail` utilisaient une version compilée (*frozen configuration*) du fichier `sendmail.cf`. Toutefois, vues les performances accrues des processeurs, l'intérêt de la version compilée est devenu nul. De plus, l'utilisation d'une version compilée génait la mise au point du fichier de configuration, et était donc fortement déconseillée.
    Ce chemin n'était pas modifiable.
  – `submit.cf`
    Fichier de configuration spécifique pour la soumission des messages.
  – `aliases`
    Les aliases en clair. Ce chemin est modifiable (option `OA` du fichier de configuration, ou argument équivalent sur la ligne de commande).
  – `aliases.db`
    Version « compilée » des aliases, générée par `newaliases` ou `sendmail -bi`. Les anciennes versions (ou les versions de `sendmail` compilées sans le support de `db`) utilisaient les fichiers `dbm` correspondants : `aliases.dir` et `aliases.pag`. La version compilée doit forcément résider dans le même répertoire que la version en clair.
  – `helpfile` (anciennement `sendmail.hf`)
    Fichier contenant l'aide affichée lors du dialogue SMTP suite à la commande `HELP`. Ce chemin est modifiable (option `OH` du fichier de configuration, ou argument équivalent sur la ligne de commande).
  – `statistics` (anciennement `sendmail.st`)
    Informations comptables. Si ce fichier existe, `sendmail` y écrit des statistiques à chaque appel d'un agent de transport (*mailer*). Ces statistiques sont affichées en clair par le programme `mailstats`. Par exemple :

```
soleil$ mailstats
Statistics from Fri Oct 18 09:08:28 1996
 M msgsfr bytes_from  msgsto   bytes_to  Mailer
 3   2308      91780K   2619      93662K  smtp
 4    159        443K      0          0K  local
========================================
 T   2467      92223K   2619      93662K
```

    Pour chaque agent (ici, local et smtp), `mailstats` affiche le numéro de l'agent dans la configuration courante (c'est-à-dire l'ordre d'apparition dans le fichier `sendmail.cf`), le nombre de messages reçus (`msgsfr`), la taille totale de tous les messages reçus (`bytes_from`), le nombre de messages transmis (`msgsto`) et la taille totale des messages transmis (`bytes_to`) par cet agent.

Ce fichier n'est pas indispensable. Si on veut utiliser ces statistiques, il faut avoir créé ce fichier au préalable par :

```
cp /dev/null /etc/mail/statistics
chmod 644 /etc/mail/statistics
```

– répertoire `/var/mail/` :
Les boîtes aux lettres des utilisateurs. Ce répertoire n'est pas connu de `sendmail`, mais il est utilisé par `/bin/mail` lors de la remise physique du message, ainsi que par les UA pour lire ces messages. Il n'est donc pas facilement modifiable.
– répertoire `/var/spool/mqueue/` :
Ce répertoire est la file d'attente des messages. Ceux-ci sont entreposés à cet endroit pendant un très court instant, ou lorsque le site destinataire ne répond pas. La file d'attente est consultable avec l'utilitaire `mailq` ou, ce qui revient au même, `sendmail -bp`. Ce chemin est modifiable (option `OQ` du fichier de configuration, ou argument équivalent sur la ligne de commande).
– répertoire `/var/spool/clientmqueue/` :
Ce répertoire est la file d'attente des messages en cours de soumission, entreposés lorsque le serveur SMTP ne répond pas. Attention aux droits : ce répertoire doit appartenir au groupe `smmsp` et être lisible et modifiable par ce groupe.
– répertoire `/var/run/` :
  – `sendmail.pid`
    Le numéro de processus de `sendmail` (sur la première ligne), ainsi que les arguments (sur la deuxième ligne) avec lesquels il est appelé par l'instance qui agit comme démon et serveur SMTP.
– répertoire `/usr/sbin/` :
  – `sendmail`
    Le programme lui-même. Ce chemin n'est généralement pas modifiable (ou modifiable à condition de laisser un lien dans le répertoire `/usr/sbin`) car nombre de programmes (les UA) codent ce chemin en dur.
– répertoire `/usr/libexec/` :
  – `mail.local`
    Ce programme ne fait pas partie de `sendmail`, à strictement parler, mais c'est l'agent qui sert à déposer les messages dans les boîtes aux lettres des utilisateurs.
– répertoire `/usr/bin/` :
  – `newaliases`
    C'est un lien sur `sendmail` pour compiler le fichier des aliases. L'appel de `newaliases` correspond en fait à l'appel de `sendmail` avec l'option `-bi`.
  – `mailq`
    C'est un lien sur `sendmail` pour visualiser le contenu de la file d'attente. L'appel de `mailq` correspond en fait à l'appel de `sendmail` avec l'option `-bp`.
  – `hoststat`
    C'est un lien sur `sendmail` (à partir de la version 8.8) pour visualiser l'état des connexions récentes. L'appel de `hoststat` correspond en fait à l'appel de `sendmail` avec l'option `-bh`.
  – `purgestat`
    C'est un lien sur `sendmail` (à partir de la version 8.8) pour effacer l'état des connexions récentes. L'appel de `purgestat` correspond en fait à l'appel de `sendmail` avec l'option `-bH`.
– répertoire `/var/log/` :
  – `mail.log`, `mail.err`, etc.
    Fichiers de logs. Voir 2.7.1, page 53.

Note importante : à partir de la version 8.9, `sendmail` est devenu très rigoureux, voire même paranoïaque, sur un certain nombre de points touchant à la sécurité. Ainsi, par exemple, `sendmail` contrôle les permissions associées tant aux fichiers qu'aux répertoires englobants : si un fichier ou un répertoire englobant est accessible en modification à tous les utilisateurs du système, `sendmail` refusera de fonctionner correctement. De même si un de ces fichiers est en réalité un lien. Il s'agit des fichiers suivants :

– fichier `sendmail.cf`

- fichiers spécifiés dans `sendmail.cf` : `aliases` et la version compilée (`.db`), `statistics`, fichiers *maps* tels que celui utilisé pour la liste noire ou la table de routages, le cas échéant, etc.
- répertoire `mqueue/`
- fichiers `.forward` des utilisateurs
- fichiers d'aliases personnels inclus (avec `:include`) dans le fichier des aliases général
- ... et tous les répertoires englobant ces fichiers.

Plutôt que de forcer `sendmail` à accepter des répertoires ouverts en écriture, mieux vaut faire un `chmod go-w ...` et un `chmod root ....`

2.4 Fichier de configuration

Le fichier de configuration de `sendmail` a pour nom `sendmail.cf`. Il contient :

- des définitions de variables
  Les variables (*macros* en terminologie `sendmail`) ont des noms formés d'un seul caractère. Les lettres minuscules ont une signification spéciale (ce sont les variables réservées de `sendmail`), les lettres majuscules sont utilisables librement. Avec la version 8 de `sendmail`, ces variables peuvent porter des noms plus longs qu'un seul caractère, à condition de les entourer par des accolades.
- des définitions de classes
  Une classe est similaire à une variable, à ceci près qu'elle peut contenir plusieurs éléments, et que l'on peut faire des tests d'appartenance ou de non appartenance. Par exemple, une classe peut contenir l'ensemble des sites UUCP à qui nous transmettons le courrier, ou encore l'ensemble des machines pour lesquelles notre machine locale agit comme dépositaire de courrier.
- des définitions d'options
  Les options modifient le comportement de `sendmail`. Il y a un grand nombre d'options (dont les noms sont une lettre minuscule ou majuscule, et que la version 8 a étendu comme pour les variables) qui vont de la spécification du fichier `aliases` jusqu'au comportement de `sendmail` en cas de serveur de noms défaillant.
- des définitions de priorités
  Le champ d'en-tête `Precedence` est traité spécialement par `sendmail` pour indiquer une priorité à donner au message. Ces priorités sont indiquées dans le message par un mot-clef, et la définition des priorités dans le fichier de configuration a pour but d'affecter une valeur numérique à chaque mot-clef.
- des définitions d'utilisateurs *fiables*
  Les utilisateurs « fiables » (*trusted users*) sont des utilisateurs ayant le droit d'appeler `sendmail` avec l'option `-f` pour changer l'identité de l'expéditeur. Ceci correspond aux programmes qui envoient des messages pour le compte d'autres utilisateurs. Par exemple, lorsque `uucp` reçoit un message depuis un autre site UUCP, il appelle `sendmail` pour router ce message. Le courrier ne doit alors pas apparaître comme venant de l'utilisateur `uucp`, mais comme venant de l'utilisateur original sur le site distant. Autrement dit, cela permet d'assurer la transmission de l'enveloppe lorsque `sendmail` est appelé par un autre programme.
- des définitions de champs d'en-tête
  Lorsqu'un message est traité, `sendmail` peut ajouter, modifier ou vérifier certains champs de l'en-tête. Si modifier les définitions des champs d'en-tête est un sport délicat qui n'est heureusement quasiment jamais nécessaire, la vérification peut s'avérer très utile pour bloquer des *spams*, comme montré en 3.7.4, page 116.
- des définitions de tables de correspondances
  Il peut arriver qu'on ait besoin d'une table de correspondance (par exemple, pour transformer une adresse `login@site` en *Prénom.Nom*`@site`). Ces tables de correspondances, avec la version 8 de `sendmail`, sont stockées en format binaire `db` ou `dbm`, suivant les options de compilation de `sendmail`.
- des définitions d'agents de transport de messages
  Les agents de transport de messages (les *mailers* en terminologie `sendmail`) ne sont *a priori* pas connus. Chaque agent de transport est défini par un nom, un programme à appeler et ses arguments, ainsi que des paramètres (taille maximum des messages, champs d'en-tête nécessaires, etc.).

Trois agents au minimum doivent être connus :
- `local` : utilisé pour la remise physique dans une boîte aux lettres ;
- `prog` : utilisé pour la remise physique à un programme ;
- `error` : il n'est pas à définir (`sendmail` le définit automatiquement), mais il peut être utilisé dans le fichier de configuration pour renvoyer un message d'erreur lorsqu'une condition d'erreur a été détectée par les règles de réécriture.

Le programme `sendmail` inclut un agent de transport spécialisé pour le protocole SMTP. Cet agent n'est pas défini implicitement, il faut le définir explicitement. Le nom de l'agent est, par convention, la chaîne `[IPC]`, et la chaîne d'arguments commence par l'« exécutable » `TCP`, toujours par convention.

– des règles de réécriture
Les règles de réécriture sont le cœur de la configuration de `sendmail`, et sont décrites dans la section suivante.

2.5 Règles de réécriture

L'essentiel du travail de `sendmail` est réalisé dans les règles de réécriture. Ces règles manipulent l'adresse par un jeu de réécritures analogues, dans le principe, à des substitutions avec `sed`.

2.5.1 Objectif

L'objectif des règles de réécriture est de :

- convertir l'adresse du destinataire figurant dans l'enveloppe en une forme canonique facilement manipulable, puis de prendre la décision de routage (c'est-à-dire choisir l'agent de transport) en fonction de l'adresse ;
- prendre chaque champ de l'en-tête contenant une adresse, et la convertir en une forme canonique, puis la réécrire en fonction de l'agent de transport désiré ;
- effectuer certaines actions, comme par exemple la validation d'un relais ou d'une adresse d'émetteur dans le cadre de la lutte anti-*spam* (voir 1.10, page 34).

Dans tous les cas, vue la diversité des adresses fournies à `sendmail`, il faut convertir ces différents formats en une forme canonique, c'est-à-dire une forme « normalisée », unique, facile à manipuler.

La réécriture en fonction de l'agent de transport sélectionné permet, par exemple, de convertir une adresse Internet en adresse UUCP si l'agent est `uucp`, ou encore de réécrire les adresses pour ne plus faire figurer que le nom de domaine et pas le nom de machine.

2.5.2 Ensemble de règles

Les règles sont regroupées en ensembles. Chaque ensemble porte un numéro (entre 0 et 29 pour les anciennes versions en général, et entre 0 et 199 pour la version 8) ou un nom (pour les versions récentes de `sendmail`).

Les ensembles 0 à 4 sont appelés dans des conditions bien précises par `sendmail`, et doivent donc être écrits pour résoudre des problèmes spécifiques (voir plus loin). Les autres ensembles ne sont pas appelés directement par `sendmail`, mais peuvent l'être (de manière analogue à des sous-programmes) par les règles 0 à 4.

Toutefois, différentes versions de `sendmail` peuvent définir d'autres ensembles (la version 8 utilise, par exemple, l'ensemble 5). Il est donc préférable d'utiliser les ensembles de numéros supérieurs à 10.

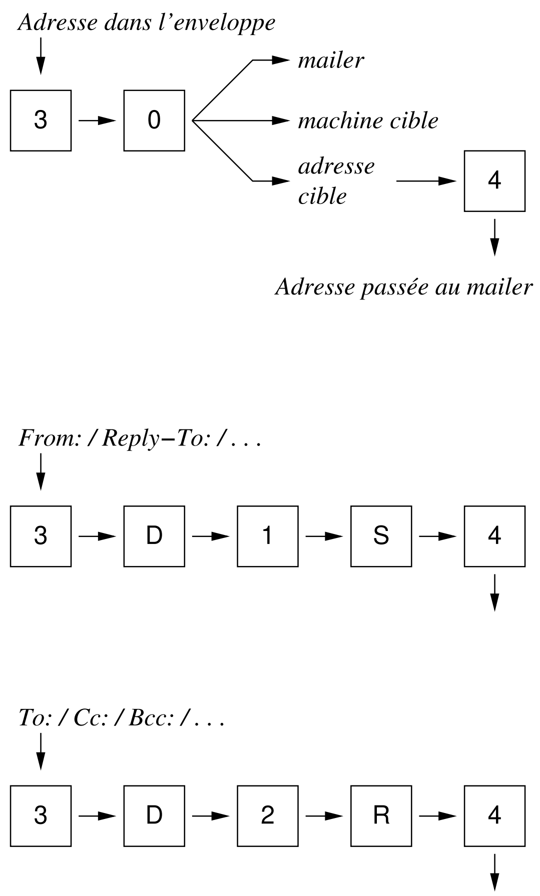

FIG. 2.3 – Règles prédéfinies dans `sendmail`

La figure 2.3 résume l'ordre de passage dans les règles :

- en tout premier lieu, l'adresse du destinataire figurant dans l'enveloppe est extraite, puis passée par les ensembles :
  - 3, pour procéder à la canonisation de l'adresse,
  - 0, pour aboutir à un triplet :
    - agent de transport (*mailer* en terminologie `sendmail`)
    - relais (*machine cible* utilisée par le *mailer*), c'est la machine avec laquelle va communiquer l'agent de transport
    - nouvelle adresse d'enveloppe (*adresse cible* utilisée sur la machine cible). La nouvelle enveloppe est une nouvelle adresse construite spécialement pour passer au relais,
  - la nouvelle adresse de l'enveloppe est ensuite passée par l'ensemble 4 pour « décanonisation », c'est-à-dire réécriture dans la forme originale ;
- ensuite, les champs de l'en-tête qui contiennent une adresse d'expéditeur (`From`, `Sender`, `Reply-To`, etc.) sont passés par les ensembles :
  - 3, pour procéder à la canonisation de l'adresse,
  - 1, pour procéder à des réécritures d'adresses communes pour tous les agents de transport. C'est par exemple dans cette règle qu'on peut traduire un nom de *login* en adresse *Prénom.Nom*,
  - *s* (un ensemble dépendant de l'agent de transport, défini en même temps que l'agent) pour procéder à des

  réécritures spécifiques. Par exemple, si un courrier est envoyé sur l'Internet et l'adresse est locale, alors cette règle ajoute le nom du site afin de ne jamais transmettre une adresse non qualifiée,
  - 4, pour décanoniser l'adresse ;

  Entre les ensembles 3 et 1, si l'adresse n'est pas qualifiée et l'agent de transport est marqué avec le *flag* `C`, alors le nom de domaine de notre machine est ajouté à l'adresse (cet ajout est représenté dans la littérature par un « pseudo-ensemble » D).
- enfin, les champs de l'en-tête qui contiennent une adresse de destinataire (`To`, `Cc`, etc.) sont passés par les ensembles :
  - 3, pour procéder à la canonisation de l'adresse,
  - 2, pour procéder à des réécritures d'adresses communes à tous les agents de transport,
  - *r* (un ensemble dépendant de l'agent de transport, défini en même temps que l'agent) pour procéder à des réécritures spécifiques,
  - 4, pour décanoniser l'adresse,

  Comme précédemment, entre les ensembles 3 et 1, si l'adresse n'est pas qualifiée et que l'agent de transport est marqué avec le *flag* `C`, le nom de domaine de notre machine est ajouté à l'adresse (cet ajout est traditionnellement représenté par un « pseudo-ensemble » D).

2.5.3 Définition des règles

Avant d'étudier le fonctionnement des règles en détail, il faut signaler quelques points importants :

- l'entrée d'un ensemble de règles est décomposée en *tokens*. L'adresse `Pierre.David@prism.uvsq.fr`, par exemple, est découpée en 9 *tokens*. Le premier est « `Pierre` », le deuxième est « `.` » et ainsi de suite jusqu'au neuvième qui est « `fr` » ;
- les comparaisons se font sans tenir compte de la distinction entre minuscules et majuscules. Ainsi, si on cherche `$*.BITNET` et si on dispose de l'adresse `toto@site.bitnet` en entrée, alors il y a concordance ;

L'algorithme de fonctionnement des ensembles de règles est très simple. Chaque règle d'un ensemble contient une partie gauche (*lhs*, pour *left hand side*) et une partie droite (*rhs*, pour *right hand side*). Les étapes de la réécriture sont les suivantes :

1. si la partie gauche correspond à l'adresse, sélectionner la règle, réécrire la règle suivant la spécification de la partie droite, puis recommencer en 1 avec la même règle ;
2. si la partie gauche ne correspond pas à l'adresse, prendre la règle suivante si elle existe, et recommencer en 1 ;
3. s'il n'y a plus de règle dans l'ensemble, il est terminé. L'adresse obtenue est le résultat de l'ensemble.

Cet algorithme définit implicitement une boucle pour chaque règle (l'adresse est réécrite tant que cela est possible). Ce fonctionnement itératif peut être éliminé pour certaines règles (opérateur `$:`). De même, le fonctionnement est séquentiel : après avoir fini d'utiliser une règle, on continue dans la suivante. Ce comportement peut également être éliminé, une règle pouvant terminer un ensemble (opérateur `$@`).

Cet algorithme est illustré sur la figure 2.4. Étant donné l'ensemble 15 comportant deux règles, et l'adresse `a%b%c%d`, la première règle (`$+%$+`) correspond et l'adresse est réécrite en `a@b%c%d`. Cette nouvelle adresse est comparée à la partie gauche de la première règle à nouveau, et elle est réécrite une deuxième fois pour donner `a@b@c%d`. Après un troisième passage dans cette règle, on obtient `a@b@c@d`. À ce stade, la première règle ne peut plus correspondre, et on passe à la seconde. Celle-ci s'applique, elle aussi, itérativement pour donner à la fin `a%b%c@d`. Comme il n'y a pas de troisième règle, l'ensemble 15 est terminé.

```
# Ensemble 15
# transforme les adresses "..%..%.." e
# et les adresses "..@..@.." en "..%..
S15
R$+%$+          $1@$2          supprime
R$+@$+@$+       $1%$2@$3       conserve
```

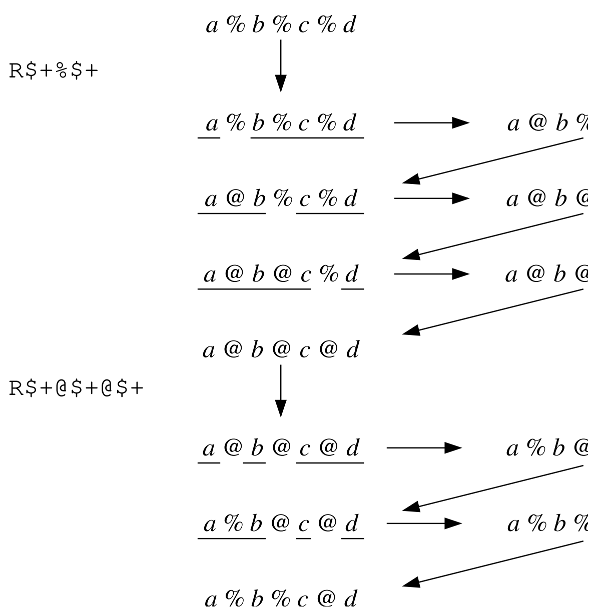

FIG. 2.4 – Fonctionnement des réécritures dans `sendmail`

2.5.4 Forme canonique

Forme canonique traditionnelle

L'ensemble de règles numéro 3 a pour but de convertir l'adresse en une forme canonique. Aucune norme ne fixe cette forme canonique, cependant la plupart des auteurs sérieux de `sendmail.cf` utilisent toujours la même (à quelques détails près). Cette forme est :

> *partie-locale* <@ *premier-relais* > *routage*

La *partie-locale* ou le *routage* peuvent être vides (mais pas en même temps).

- la *partie-locale* contient le nom (`toto`) de l'utilisateur dans l'adresse (`toto@site` par exemple) ;
- le *premier-relais* contient l'adresse à utiliser (`site` pour reprendre l'exemple précédent) ;
- le *routage* est utilisé lorsqu'on a affaire à une adresse avec routage explicite (*source-route*). Dans le cas de l'adresse <site1,site2:user@site3>, la forme canonique est :

  ```
  <@site1>:site2:user@site3
  ```

  Autrement dit, le premier « : » est là pour rappeler qu'il s'agit d'une adresse avec routage explicite, il n'y a pas de partie locale, et le relais est la première machine à contacter.

Dans le cas où l'adresse n'est pas qualifiée, la canonisation est identique à l'adresse.

Forme canonique du kit Jussieu

Un des principes du kit de Jussieu étant de classer les adresses en trois catégories, la forme canonique est quasiment identique, à ceci près que :

- le *premier-relais* est suffixé par `.LOCAL`, `.INTERNE` ou `.EXTERNE` ;
- si l'adresse n'est pas initialement qualifiée, la forme canonique est `user<@.LOCAL>` (c'est-à-dire avec un seul suffixe).

2.5.5 Règles anti-spam

Depuis la version 8.8, `sendmail` reconnaît quatre nouveaux ensembles de règles[1] afin d'effectuer des vérifications. Si un ensemble aboutit au mailer `error`, la vérification a échoué et le courrier est refusé. Si aucun ensemble n'aboutit au mailer `error`, toutes les vérifications ont réussi, le courrier est donc accepté.

Ces quatre ensembles sont :

- `check_relay` : cet ensemble est appelé lors de l'ouverture de la session SMTP, afin de valider le nom et l'adresse IP du client. Le paramètre est un couple « *nom* `$|` *adresse IP* » (le séparateur est `$|`). Si la partie « *nom* » est encadrée par des crochets, il s'agit de l'adresse IP car le nom n'a pas pu être trouvé.

[1] Les versions ultérieures en admettent encore plus.

Les vérifications typiques qui peuvent être effectuées dans cet ensemble sont la présence du nom dans le DNS (refus des adresses IP numériques), et la présence du client dans une liste noire.
Avec la version 8.8, le code d'erreur retourné est toujours `550 Access denied`, quels que soient le message et le code spécifiés lors de l'appel du mailer `error`. Depuis la version 8.9, ce n'est plus le cas : le message et le code sont paramétrables.
- `check_mail` : cet ensemble est appelé lors de la réception de l'adresse d'expéditeur (mot-clef `MAIL FROM` dans l'échange SMTP). Le paramètre est l'adresse (enveloppe) fournie par le client. Il faut noter que les macros `client_name` et `client_addr` sont définies avec le nom et l'adresse IP du client.
Les vérifications typiques qui peuvent être effectuées dans cet ensemble sont la validité de l'adresse (adresse complète, adresse absente de la liste noire), ou la conformité de l'adresse par rapport au client SMTP (pour refuser une adresse d'émetteur locale lorsque le client est distant).
Le code et le message d'erreur sont spécifiés dans l'appel du mailer `error`.
- `check_rcpt` : cet ensemble est appelé à chaque fois que le client spécifie une adresse de destinataire (mot-clef `RCPT TO` dans l'échange SMTP). Le paramètre est l'adresse (enveloppe) fournie par le client. Il faut noter que les macros `client_name` et `client_addr` sont définies avec le nom et l'adresse IP du client.
Les vérifications typiques qui peuvent être effectuées dans cet ensemble sont la conformité de l'adresse par rapport au client SMTP (pour refuser les adresses de destinataire distantes lorsque le client est distant).
Le code et le message d'erreur sont spécifiés dans l'appel du mailer `error`.
- `check_compat` : cet ensemble est appelé lorsque `sendmail` a collecté le courrier dans sa totalité (c'est-à-dire après le point terminal de `DATA` dans l'échange SMTP, ou lorsque le fichier a été totalement lu si `sendmail` est appelé localement). Le paramètre est un couple « *émetteur* `$|` *destinataire* » (le séparateur est `$|`).
Du fait de l'appel tardif de cette règle, s'il y a erreur, celle-ci ne peut pas être signalée au client SMTP, mais le message sera quand même rejeté.
Les vérifications typiques qui peuvent être effectuées dans cet ensemble sont la cohérence des deux adresses, pour empêcher de servir de relais involontaire.

Le code d'erreur renvoyé a une grande importance : s'il est de la forme 5*xy*, l'erreur est définitive et le client doit retourner le message à l'émetteur ; s'il est de la forme 4*xy*, l'erreur est temporaire, et le client devra refaire une tentative ultérieure.

Quelle valeur choisir ? Chaque solution a ses avantages et ses inconvénients :

- renvoyer le code 5*xy* est définitif, il faut être sûr que la raison du rejet est fiable. Cela n'est pas le cas, par exemple, si un nom n'a pas été trouvé dans le DNS, car le *resolver* n'a peut-être pas disposé du temps nécessaire, notamment si le client SMTP est de l'autre côté de la planète ;
- renvoyer le code 4*xy* peut « géner » le *spammeur* en lui faisant réïtérer sa requête. L'inconvénient est qu'un utilisateur honnête, mais dont l'agent utilisateur est mal configuré, n'apprendra qu'au bout de quelques jours que son courrier est rejeté. Toutefois, il est fortement déconseillé d'utiliser ce code.

Le kit Jussieu permet d'employer les deux valeurs, suivant la fiabilité de la cause d'erreur (recherche dans le DNS ou non).

2.5.6 Filtrage spécifique sur les champs d'en-tête

Depuis la version 8.9, `sendmail` offre la possibilité d'analyser des champs spécifiques de l'en-tête. Il devient ainsi faisable de refuser le message si le sujet contient un motif, ou si les champs de trace (`Received`) contiennent le nom ou l'adresse d'une machine honnie.

Par exemple, pour filtrer un certain sujet, on utilisera :

```
# Verifier le sujet des message
HSubject:          $>verifier_sujet

# Le motif que l'on cherche
DX ILOVEYOU

Sverifier_sujet
# Rechercher si on trouve le motif, avec ou sans les préfixes habituels
R $X $*            $#error $: 550 This message contains the ILOVEYOU virus
R Re: $X           $#error $: 550 This message contains the ILOVEYOU virus
R Fw: $X           $#error $: 550 This message contains the ILOVEYOU virus
R Tr: $X           $#error $: 550 This message contains the ILOVEYOU virus
```

Un autre exemple de filtrage, sur les champs `Received`, est exposé en 3.7.4, page 116.

2.6 Utilisation de LDAP

Le support de LDAP est intégré à `sendmail` depuis la version 8.9, et s'est bonifié au fil du temps. Le principal intérêt de l'utilisation de LDAP avec `sendmail` est d'accéder à une base de données centralisée de toutes les informations utiles :

– aliases, ou conversions de noms de messagerie en logins Unix
– revaliases, ou conversions inverses
– liste noire
– routage des courriers
– etc.

Le programme `sendmail` réalise l'accès à un serveur LDAP de manière très similaire à l'utilitaire `ldapsearch` fourni avec OpenLDAP. On peut spécifier les paramètres d'accès au serveur (adresse, port, DN[2] de base pour les recherches), et plus généralement le filtre utilisé pour obtenir l'information.

Si cette manière de faire autorise n'importe quel schéma pour la base LDAP, `sendmail` privilégie toutefois un schéma particulier pour l'accès aux informations, spécifié dans le fichier `sendmail.schema` accompagnant les versions actuelles de `sendmail` :

[2] Distinguished Name

Type	Attributs requis	Attributs optionnels
sendmailMTA	objectClass	sendmailMTACluster sendmailMTAHost Description
sendmailMTAMap	objectClass sendmailMTAMapName	sendmailMTACluster sendmailMTAHost Description
sendmailMTAObject	objectClass sendmailMTAMapName sendmailMTAKey sendmailMTAMapValue	sendmailMTACluster sendmailMTAHost Description
sendmailMTAAlias	objectClass	sendmailMTAAliasGrouping sendmailMTACluster sendmailMTAHost Description
sendmailMTAAliasObject	objectClass sendmailMTAKey sendmailMTAAliasValue	sendmailMTAAliasGrouping sendmailMTACluster sendmailMTAHost Description
sendmailMTAClass	objectClass sendmailMTAClassName sendmailMTAClassValue	sendmailMTACluster sendmailMTAHost Description

Par défaut, `sendmail` limite sa recherche aux seules entrées dont l'attribut `sendmailMTAHost` correspond au nom de la machine, tel que défini avec la macro `$j`, ou l'attribut `sendmailMTACluster` correspond au nom défini par la macro `${sendmailMTACluster}`. Cette dernière possibilité permet à plusieurs machines de partager un ensemble d'aliases, par exemple, sans avoir à les répéter pour chaque machine.

Les types d'objets cités dans le tableau ci-dessus correspondent aux différentes utilisations dans le fichier de configuration.

– l'objet `sendmailMTAAliasObject` sert à représenter un alias. Par exemple, pour définir une liste de diffusion et un alias de correspondance Nom.Prénom, il faut spécifier les entrées LDIF suivantes :

```
dn: sendmailMTAKey=maliste, dc=domaine, dc=fr
objectClass: sendmailMTA
objectClass: sendmailMTAAlias
objectClass: sendmailMTAAliasObject
sendmailMTAAliasGrouping: aliases
sendmailMTAHost: serveur.domaine.fr
sendmailMTAKey: maliste
sendmailMTAAliasValue: machin@labas.fr
sendmailMTAAliasValue: truc@ailleurs.org
sendmailMTAAliasValue: bidule

dn: sendmailMTAKey=Jean.Breille, dc=domaine, dc=fr
objectClass: sendmailMTA
objectClass: sendmailMTAAlias
objectClass: sendmailMTAAliasObject
sendmailMTAAliasGrouping: aliases
sendmailMTAHost: serveur.domaine.fr
sendmailMTAKey: Jean.Breille
sendmailMTAAliasValue: jb
```

Ce type d'objet peut également servir à spécifier les correspondances inverses (revaliases). Par exemple :

```
dn: sendmailMTAKey=Jean.Breille, dc=domaine, dc=fr
objectClass: sendmailMTA
objectClass: sendmailMTAAlias
objectClass: sendmailMTAAliasObject
sendmailMTAAliasGrouping: revaliases
sendmailMTAHost: serveur.domaine.fr
sendmailMTAKey: Jean.Breille
sendmailMTAAliasValue: jb
```

– l'objet `sendmailMTAObject` sert à représenter plus généralement une association (clef, valeur). Par exemple, une table de routage du Kit Jussieu (voir voir 3.6.10, page 87 pour l'exemple) pourrait être représentée par :

```
dn: sendmailMTAKey=frmug.fr.net, dc=domaine, dc=fr
objectClass: sendmailMTA
objectClass: sendmailMTAMap
objectClass: sendmailMTAMapObject
sendmailMTAHost: serveur.domaine.fr
sendmailMTAMapName: routages
sendmailMTAKey: frmug.fr.net
sendmailMTAMapValue: uucp.frmug

dn: sendmailMTAKey=machine-speciale.prism.uvsq.fr, dc=domaine, dc=fr
objectClass: sendmailMTA
objectClass: sendmailMTAMap
objectClass: sendmailMTAMapObject
sendmailMTAHost: serveur.domaine.fr
sendmailMTAMapName: routages
sendmailMTAKey: machine-speciale.prism.uvsq.fr
sendmailMTAMapValue: smtp.[machine-speciale.prism.uvsq.fr]
```

Dans cet exemple, l'attribut `sendmailMTAMapName` correspond au nom de la *map* utilisée dans le fichier de configuration `sendmail.cf`.

– l'objet `sendmailMTAClass`, quant à lui, sert à initialiser des classes ou des macros au lancement de `sendmail`.

Comme précisé ci-dessus, toutefois, ce schéma n'est qu'indicatif. Dans la mesure où tous les filtres sont paramétrables, il est tout à fait possible d'utiliser n'importe quel autre schéma. Seule la définition du fichier des aliases est plus simple si on choisit d'utiliser le schéma par défaut.

2.7 Mise au point

Avant d'installer un nouveau fichier de configuration, il faut le tester. La démarche est décrite ci-dessous.

1. Tester les règles de réécriture sans installer le nouveau fichier de configuration. Cette étape ne nécessite aucun privilège.
2. Tenter d'expédier un courrier sans installer le nouveau fichier de configuration, en appelant directement `sendmail`. Il faut être `root` pour tester ceci.

3. Mettre en place le serveur sur un port autre que le port réservé à SMTP (port 25). Cela permet de tester l'arrivée des messages via le *mailer* `smtp` sans pour autant perturber l'arrivée normale du courrier. Il faut être `root` pour tester ceci.
4. Installer le nouveau fichier de configuration et tester l'expédition de courriers sans interrompre le programme `sendmail` qui tourne éventuellement en mode `-bd`. Il faut être `root` pour installer le nouveau fichier, mais pas forcément pour expédier des courriers.
5. Arrêter le programme qui tourne en mode `-bd` (s'il y en a un) et le relancer pour qu'il tienne compte du nouveau fichier de configuration, puis tester l'expédition depuis une autre machine. Il faut être `root` pour arrêter et relancer `sendmail`, mais pas pour tester l'expédition depuis une nouvelle machine.

Les sections ci-après détaillent les différents outils disponibles pour tester une nouvelle configuration. Mais il n'est pas inutile de rappeler que `sendmail` peut être appelé par deux canaux :

– par les UA locaux (ou par des programmes tels que `uucp`) qui lancent une nouvelle copie du programme `sendmail`. Ceci signifie que la nouvelle configuration est relue lors de l'expédition de tout courrier local ;
– par une connexion SMTP depuis un site distant. Cette connexion se fait avec le programme `sendmail` lancé avec l'option `-bd` au démarrage du système[3]). Dans ce cas, le fichier de configuration est lu au moment où `sendmail` est démarré, il n'est plus jamais relu par la suite.

Autrement dit, si vous installez un nouveau fichier de configuration, il ne sera pris en compte que par les nouvelles invocations de `sendmail`, c'est-à-dire par les courriers émis depuis la machine locale, et par les courriers reçus depuis des sites distants si vous avez relancé la version qui tourne avec l'option `-bd`.

2.7.1 Syslog

La première chose à faire est de configurer l'utilitaire `syslogd`, qui est le dispositif central d'affichage et de journalisation des messages d'erreur. Le programme `sendmail`, comme tout programme bien conçu, utilise `syslogd` et est capable de sortir un grand nombre d'informations par ce biais.

La première étape est de configurer correctement le fichier `/etc/syslog.conf`[4]. Il faut en particulier vérifier que les lignes :

```
mail.debug          /usr/adm/mail.log
mail.err            /usr/adm/mail.err
```

(ou équivalentes) sont présentes. Les deux fichiers spécifiés doivent exister au préalable, et `syslogd` doit être arrêté et redémarré lorsque ce fichier est modifié. Les versions récentes de `syslogd` relisent ce fichier lorsqu'on leur envoie le signal `SIGHUP`, ce qui évite d'avoir à l'arrêter et le redémarer.

Le fichier `mail.log` contient des lignes comme :

```
Oct 18 09:30:51 soleil sendmail[13749]: RAA13749: from=pda,
    size=56, class=0, pri=30056, nrcpts=1,
    msgid=<199406131525.RAA13749@soleil.uvsq.fr>,
    relay=pda@localhost
```

[3] On peut aussi ne pas lancer `sendmail` avec cette option au démarrage, si la machine ne doit jamais recevoir un courrier, voir le kit de configuration.

[4] Il faut savoir que bon nombre de constructeurs livrent un fichier `syslog.conf` inutilisable tel quel.

```
Oct 18 09:30:55 soleil sendmail[13749]: RAA13749: to=pda@jussieu.fr,
    ctladdr=pda (10001/10000), delay=00:00:08,
    mailer=smtp, relay=shiva.jussieu.fr.  [134.157.0.129],
    stat=Sent (RAA13994 Message accepted for delivery)
```

Dans cet exemple (issu de l'exemple d'expédition directe de la section 2.7.4 en page 58), `sendmail` a généré deux lignes (découpées ici pour la lisibilité). La première décrit le message tel qu'il a été reçu, la deuxième tel qu'il a été envoyé.

Le fichier `mail.err`, quant à lui, contient des informations qu'il est bon d'investiguer. Il peut contenir également des informations utiles au niveau sécurité (détection d'un *cracker* potentiel).

Une bonne pratique consiste à utiliser `tail -f /usr/adm/mail.log` dans une fenêtre, pour afficher les informations au fur et à mesure que les messages transitent.

2.7.2 Option test

Il est souhaitable de tester une nouvelle configuration avec l'option `-bt`. Par exemple :

```
soleil$ /usr/lib/sendmail -bt -C $HOME/sendmail.cf.new
```

L'option `-C` spécifie un fichier de configuration différent du fichier de configuration par défaut. L'option `-bt` demande à `sendmail` d'utiliser le mode test. L'affichage est alors :

```
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
>
```

Il faut alors entrer un ou plusieurs ensembles de règles, puis une adresse à tester. Par exemple :

```
> 3 pda@uvsq.fr
rewrite: ruleset  3   input: pda @ uvsq . fr
rewrite: ruleset 19   input: pda < @ uvsq . fr >
rewrite: ruleset 19 returns: pda < @ uvsq . fr . LOCAL >
rewrite: ruleset  3 returns: pda < @ uvsq . fr . LOCAL >
```

Dans cet exemple, la règle 3 est appelée avec comme entrée `pda@uvsq.fr`. La règle 3 appelle la règle 19 (elle est programmée comme ça dans `sendmail.cf.new`), qui renvoie l'adresse `pda<@uvsq.fr.LOCAL>` après toutes les réécritures. Cette nouvelle adresse est également l'adresse que renvoie la règle 3, c'est donc l'adresse canonisée.

On peut aussi enchaîner plusieurs ensembles :

```
> 3,0 pda@prism.uvsq.fr
rewrite: ruleset  3   input: pda @ prism . uvsq . fr
rewrite: ruleset 19   input: pda < @ prism . uvsq . fr >
rewrite: ruleset 19 returns: pda < @ prism . uvsq . fr . INTERNE >
```

```
rewrite: ruleset  3 returns: pda < @ prism . uvsq . fr . INTERNE >
rewrite: ruleset  0   input: pda < @ prism . uvsq . fr . INTERNE >
rewrite: ruleset  0 returns: $# smtp $@ [ mailhost . prism . uvsq . fr ]
             $: pda < @ prism . uvsq . fr . INTERNE >
```

Dans cet exemple, l'adresse est passée par la règle 3, puis l'adresse une fois canonisée (c'est-à-dire sous la forme `pda<@prism.uvsq.fr.INTERNE>` dans ce cas) est transmise à la règle 0. Le but de celle-ci étant de déterminer l'agent de transport, la règle 0 se termine par un triplet (agent de transport, relais, nouvelle enveloppe). L'agent de transport est ici `smtp`, le relais est `[mailhost.prism.uvsq.fr]` ce qui signifie, en version 8, qu'il ne faut pas consulter les MX puisque le nom est entre crochets. La nouvelle enveloppe correspond à l'adresse originale dans ce cas.

Pour tester la réécriture de l'adresse de l'expéditeur, on peut faire, en considérant que la règle 11 est la règle de réécriture des adresses d'expéditeur pour l'agent de transport SMTP :

```
> 3,1,11,4 pda
rewrite: ruleset  3   input: pda
rewrite: ruleset 19   input: pda
rewrite: ruleset 19   input: pda < @ . LOCAL >
rewrite: ruleset 19 returns: pda < @ . LOCAL >
rewrite: ruleset 19 returns: pda < @ . LOCAL >
rewrite: ruleset  3 returns: pda < @ . LOCAL >
rewrite: ruleset  1   input: pda < @ . LOCAL >
rewrite: ruleset 17   input: pda < @ . LOCAL >
rewrite: ruleset 17 returns: Pierre . David < @ . LOCAL >
rewrite: ruleset  1 returns: Pierre . David < @ . LOCAL >
rewrite: ruleset 11   input: Pierre . David < @ . LOCAL >
rewrite: ruleset 16   input: Pierre . David < @ . LOCAL >
rewrite: ruleset 16 returns: Pierre . David < @ uvsq . fr . LOCAL >
rewrite: ruleset 11 returns: Pierre . David < @ uvsq . fr . LOCAL >
rewrite: ruleset  4   input: Pierre . David < @ uvsq . fr . LOCAL >
rewrite: ruleset  4 returns: Pierre . David @ uvsq . fr
```

2.7.3 Niveaux de debug

Le mode `-bt` vu précédemment est suffisant pour détecter la majeure partie des problèmes. Toutefois, il existe certains cas où il faut faire appel à des diagnostics plus élaborés. Le programme `sendmail` est très complet dans ce domaine, puisqu'il est capable de tracer un grand nombre d'actions.

Il y a un grand nombre de niveaux de debug, mais les plus immédiatement utiles sont :

Niveau	Signification
21.1	affiche l'adresse en cause en cas de récursion infinie
21.2	affiche les adresses à l'entrée et à la sortie d'un ensemble, ainsi que les variables demandant une évaluation lors de l'expansion
21.3	affiche les adresses lors des appels aux ensembles
21.4	affiche les adresses réécrites par les ensembles
21.10	affiche les règles non utilisées
21.15	visualise le comportement de l'algorithme
21.35	affiche la reconnaissance de l'adresse en fonction de la partie gauche des règles
21.36	affiche en détail la reconnaissance de l'adresse

En pratique, le niveau `-d21.2` permet de suivre l'envoi d'un courrier et le niveau `-d21.15` permet d'analyser le fonctionnement des règles de réécriture. Par exemple[5] :

```
soleil# /usr/lib/sendmail -C $HOME/sendmail.cf.new -d21.2 pda
From: pda
To: pda

essai
^D
```

provoque l'affichage de

```
rewrite: ruleset  3   input: pda
rewrite: ruleset 19   input: pda
rewrite: ruleset 19   input: pda < @ . LOCAL >
rewrite: ruleset 19 returns: pda < @ . LOCAL >
rewrite: ruleset 19 returns: pda < @ . LOCAL >
rewrite: ruleset  3 returns: pda < @ . LOCAL >
rewrite: ruleset  0   input: pda < @ . LOCAL >
rewrite: ruleset  0 returns: $# local $: pda
rewrite: ruleset  4   input: pda
rewrite: ruleset  4 returns: pda

rewrite: ruleset  3   input: pda
rewrite: ruleset 19   input: pda
rewrite: ruleset 19   input: pda < @ . LOCAL >
rewrite: ruleset 19 returns: pda < @ . LOCAL >
rewrite: ruleset 19 returns: pda < @ . LOCAL >
rewrite: ruleset  3 returns: pda < @ . LOCAL >
rewrite: ruleset  1   input: pda < @ . LOCAL >
rewrite: ruleset 17   input: pda < @ . LOCAL >
rewrite: ruleset 17 returns: Pierre . David < @ . LOCAL >
rewrite: ruleset  1 returns: Pierre . David < @ . LOCAL >
rewrite: ruleset  4   input: Pierre . David < @ . LOCAL >
rewrite: ruleset  4 returns: Pierre . David
```

[5]Cet exemple utilise l'expédition directe, voir section 2.7.4.

```
rewrite: ruleset  3   input: pda
rewrite: ruleset 19   input: pda
rewrite: ruleset 19   input: pda < @ . LOCAL >
rewrite: ruleset 19 returns: pda < @ . LOCAL >
rewrite: ruleset 19 returns: pda < @ . LOCAL >
rewrite: ruleset  3 returns: pda < @ . LOCAL >
rewrite: ruleset  0   input: pda < @ . LOCAL >
rewrite: ruleset  0 returns: $# local $: pda
rewrite: ruleset  2   input: pda
rewrite: ruleset  2 returns: pda
rewrite: ruleset 20   input: pda
rewrite: ruleset 20 returns: pda
rewrite: ruleset  4   input: pda
rewrite: ruleset  4 returns: pda
```

La première série correspond à la sélection de l'agent de transport à l'aide de l'adresse figurant dans l'enveloppe (c'est-à-dire l'argument de `sendmail`). La deuxième série montre la réécriture de l'adresse de l'expéditeur et la troisième l'adresse du destinataire.

Le niveau `-d21.15` détaille très finement le fonctionnement de l'algorithme des règles de réécriture. Par exemple :

```
soleil$ /usr/lib/sendmail -C $HOME/sendmail.cf.new -bt -d21.15
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 1 pda<@.LOCAL>
rewrite: ruleset  1   input: pda < @ . LOCAL >
-----trying rule: $*
-----rule matches: $: $> 17 $1
$1: 7b033de4="pda" 7b033de8="<" 7b033dea="@" 7b033dec="." 7b033dee="LOCAL" 7b033df4=">"
-----callsubr 17
rewrite: ruleset 17   input: pda < @ . LOCAL >
-----trying rule: $* < @ $+ . LOCAL > $*
----- rule fails
-----trying rule: $* < @ $* . LOCAL > $*
-----rule matches: $@ $( revaliases $1 $) < @ $2 . LOCAL > $3
$1: 7b033de4="pda"
$2:
$3:
rewritten as: Pierre . David < @ . LOCAL >
rewrite: ruleset 17 returns: Pierre . David < @ . LOCAL >
rewritten as: Pierre . David < @ . LOCAL >
rewrite: ruleset  1 returns: Pierre . David < @ . LOCAL >
>
```

Pour ne pas trop alourdir l'exemple, l'ensemble de règles numéro 1 est directement appelé avec une adresse en forme canonique (`pda<@.LOCAL>`). Cet ensemble 1 reconnaît le motif `$*`, et donc appelle l'ensemble 17. Dans ce nouvel ensemble, la première règle échoue, la deuxième réussit, et la réécriture conduit à l'adresse finale.

Il existe de nombreux autres catégories et niveaux de debug. Les catégories sont listées dans les sources de `sendmail` (fichier `src/TRACEFLAGS`).

2.7.4 Expédition directe

Pour tester l'expédition d'un courrier sans installer un nouveau fichier de configuration, on peut utiliser directement `sendmail` sans passer par un UA. On peut ainsi utiliser l'option `-C` pour spécifier un autre fichier de configuration que le fichier par défaut. Cette manipulation permet de tester à la fois le fichier de configuration (mais il vaut mieux avoir déjà testé les règles de réécriture au préalable), ainsi que l'émission via les différents *mailers* définis dans le fichier de configuration.

L'interface est assez frustre, mais suffisante pour tester la plupart des cas. Par exemple[6], si on désire tester le *mailer* `local`, on fera :

```
soleil# /usr/lib/sendmail -v -C $HOME/sendmail.cf.new pda
From: pda
To: pda
Subject: essai

essai numero 1
^D
pda... Sent
```

Le message est tapé sur l'entrée standard, en comprenant les champs d'en-tête (seuls `From:` et `To:` sont véritablement indispensables), une ligne vide, et ensuite le corps du message. Lorsqu'on rentre le Control-D habituel, le message est envoyé et `sendmail` affiche `pda... Sent`.

Pour tester le *mailer* SMTP, on fera :

```
soleil# /usr/lib/sendmail -v -C $HOME/sendmail.cf.new pda@jussieu.fr
From: pda
To: pda@jussieu.fr
Subject: essai

essai numero 2
^D
pda@jussieu.fr... Connecting to shiva.jussieu.fr. (smtp)...
220-shiva.jussieu.fr Sendmail 8.12.1/jtpda-5.4 ready at Fri, 23 Nov 2001
09:30:15 +0100
220 ESMTP spoken here
>>> EHLO soleil.uvsq.fr
250-shiva.jussieu.fr Hello soleil.uvsq.fr, pleased to meet you
250-SIZE
250-8BITMIME
250 HELP
>>> MAIL From:<Pierre.David@uvsq.fr> SIZE=56
250 <Pierre.David@uvsq.fr>... Sender ok
>>> RCPT To:<pda@jussieu.fr>
250 <pda@jussieu.fr>... Recipient ok
>>> DATA
354 Enter mail, end with "." on a line by itself
>>> .
250 RAA13994 Message accepted for delivery
```

[6] Il faut utiliser le compte `root` car `sendmail` doit accéder à la file d'attente.

```
pda@jussieu.fr... Sent (RAA13994 Message accepted for delivery)
Closing connection to shiva.jussieu.fr.
>>> QUIT
221 shiva.jussieu.fr closing connection
```

Cette fois-ci, après la saisie du Control-D de fin, `sendmail` affiche le dialogue SMTP avec le site distant. Chaque ligne reçue par le serveur SMTP distant (ici `shiva.jussieu.fr`) contient un code numérique en début de ligne, donc est facilement identifiable. Chaque ligne émise par notre site commence par `>>>` et est donc également facilement identifiable.

2.7.5 Tests sur un port TCP différent

Par rapport au test précédent, ce test permet de vérifier la réception des messages via SMTP sans perturber l'arrivée normale des messages. Pour cela, il faut démarrer `sendmail` avec les options suivantes :

```
soleil# /usr/lib/sendmail -C $HOME/sendmail.cf.new -bd -oOport=26
```

Le serveur SMTP est donc démarré sur le port TCP numéro 26. Il suffit maintenant de faire `telnet localhost 26` pour entamer le dialogue SMTP (voir 1.5.1, page 14) et envoyer un message vers un utilisateur local.

2.7.6 Réflecteurs

Lorsqu'on pense que l'émission et la réception fonctionnent bien, il faut vérifier les en-têtes de vos messages, tels que vos correspondants vont les voir. Pour cela, il y a deux méthodes.

La première est la plus simple, mais elle ne donne pas exactement ce que vos correspondants verront. Elle consiste à envoyer un courrier à `vous%votre-adresse.fr@autre-domaine.fr`. Il faut noter que cette méthode ne fonctionne pas si le site servant de relais (`autre-domaine.fr` dans cet exemple) a mis en place des règles anti-*spam* sérieuses.

La deuxième consiste à envoyer un message à l'un des nombreux réflecteurs de courrier, comme par exemple `echo@univ-rennes1.fr`. Ces réflecteurs reçoivent votre message, copient l'en-tête dans un nouveau message et vous envoient le résultat. Cette méthode est plus rigoureuse, car les en-têtes tels qu'ils sont vus par le réflecteur (c'est-à-dire par vos correspondants) sont copiés dans le corps du nouveau message et ne courront donc pas le risque d'être réécrits par votre `sendmail`.

Chapitre 3

Kit de configuration

Ce chapitre décrit le kit de configuration de Jussieu. L'objectif de ce kit est de fournir un fichier de configuration (`sendmail.cf`) général (c'est-à-dire pour tous les cas « normaux » des d'organismes, qu'ils soient commerciaux ou du monde de l'éducation et de la recherche, voire même pour des cas plus exotiques) à partir d'une description de haut niveau, sans avoir besoin de « mettre ses mains dans le cambouis »...

3.1 Disponibilité

Le kit de Jussieu et la présente documentation sont disponibles en ftp anonyme à l'adresse :

```
ftp://ftp.jussieu.fr/jussieu/sendmail/kit/
```

Des versions binaires de `sendmail` 8 sont disponibles sur :

```
ftp://ftp.jussieu.fr/jussieu/sendmail/bin/
```

Les sources de la version la plus à jour de `sendmail` sont disponibles sur :

```
ftp://ftp.jussieu.fr/jussieu/sendmail/src/
```

Enfin, deux listes de diffusion peuvent intéresser les utilisateurs de ce kit :

– `kit-jussieu@jussieu.fr`
  Cette liste de diffusion est dédiée plus particulièrement aux discussions sur ce kit (mise en œuvre, évolutions, etc.). Le trafic y est très faible.
  Pour vous abonner, faites :

  `echo SUB kit-jussieu` *votre-prénom votre-nom* `| mail listserv@jussieu.fr`

– `smtp-fr@pasteur.fr`
  Cette liste est dédiée à tous les aspects de la messagerie SMTP dans un environnement francophone (c'est-à-dire pas particulièrement `sendmail`, mais tous problèmes multi-plateformes, MIME et évolutions, etc.). Le trafic est également assez faible.
  Pour vous abonner, faites :

  `echo SUB smtp-fr` *votre-prénom votre-nom* `| mail listserv@pasteur.fr`

3.2 Historique

Le kit de Jussieu a été conçu au début 1991 par Jacky Thibault et l'auteur de ce document pour répondre à un besoin précis : fournir aux administrateurs des réseaux du campus Jussieu un outil facile à utiliser. Il faut savoir que les administrateurs des réseaux du campus Jussieu, comme dans beaucoup de sites universitaires, varient de l'administrateur le plus expérimenté et déjà spécialiste de `sendmail` jusqu'au thésard non informaticien auquel on a confié, souvent contre son gré, la mission d'administrer une machine Unix. Il fallait donc un outil utilisable par ces deux extrêmes aussi bien que par tous les niveaux intermédiaires.

Pour cela, nous avons choisi de dédier une machine pour recevoir et émettre tout le courrier externe du campus. Cette machine est supposée avoir une administration plus poussée que les autres, et c'est surtout la seule à avoir une gestion sophistiquée du courrier. Cette machine redistribue le courrier dans les laboratoires, via un relais de laboratoire.

Au niveau d'un laboratoire, une machine doit servir de relais (c'est ce que nous avons appelé le *mailhost*), et les autres sont des feuilles (dans l'arbre de distribution du courrier). Le *mailhost* redistribue éventuellement le courrier à l'intérieur du laboratoire, ou le garde localement si l'administrateur préfère une gestion centralisée du courrier. La configuration des machines d'un laboratoire (*mailhost* ou feuille) n'est pas aussi complexe que la configuration du relais du campus.

Cette simplification a rendu possible la génération automatique de fichiers `sendmail.cf` par les administrateurs des laboratoires.

Pour ce faire, nous avons défini un schéma basé sur un script (le *configurateur*) et un fichier de variables adaptées à la machine à configurer. Un administrateur édite son fichier de variables, très simple, et lance le configurateur pour obtenir son fichier de configuration.

Le fichier de variables est en réalité un *script shell* exécuté par le configurateur. Le configurateur contient lui-même le fichier `sendmail.cf`, mais il utilise le préprocesseur du langage C pour l'extraire, ce qui permet d'utiliser des directives comme `#define` ou `#ifdef`.

La création de l'Université de Versailles - Saint Quentin en Yvelines a donné l'occasion de retirer tout ce qui était spécifique à Jussieu et de paramétrer le nom de domaine.

La première version contenait en fait deux versions séparées pour les deux catégories de machines (*mailhost* ou feuille). Lors du passage à V8, de sérieuses modifications ont été entreprises, notamment la fusion des deux versions en une seule.

Devant le nombre de demandes provenant d'autres sites, souhaitant reproduire un modèle comparable, mais non forcément identique à celui de Jussieu, une généralisation du configurateur devenait indispensable.

La version actuelle est le résultat d'une généralisation encore beaucoup plus poussée. D'autres sites (universités, entreprises, individuels), de configuration et/ou de taille non comparables (y compris dans d'autres pays), ont adopté ce configurateur avec succès.

3.3 Fichiers `sendmail.cf` et `submit.cf`

Depuis la version 8.12, `sendmail` utilise deux fichiers de configuration différents (voir 2.2.3, page 39) suivant le mode opératoire : soumission d'un message, (`submit.cf`) ou relais de messagerie (`sendmail.cf`).

Le kit génère un fichier pour le mode « relais de messagerie ».

Pour obtenir un fichier pour le mode « soumission d'un message », le plus simple est de prendre le fichier `submit.cf` qui accompagne la distribution de `sendmail`.

3.4 Modes d'utilisation

Le kit de Jussieu permet, bien sûr, de configurer une station. Mais il ajoute également une fonctionnalité nouvelle, qui devrait intéresser la plupart des sites d'enseignement et de recherche, en introduisant une configuration à deux niveaux, dont l'un décide de la politique de l'établissement.

3.4.1 Mode simple

Dans le mode le plus simple, une fois que le kit a été obtenu, il suffit :

- d'éditer un fichier (`toto.config` par exemple) contenant toutes les valeurs des variables décrites dans la suite de ce chapitre ;
- de lancer le configurateur par :

  ```
  ./configurateur vide/regles.vide toto.config > sendmail.cf
  ```

  le fichier `vide/regles.vide` (livré avec le kit) est un fichier qui ne contient aucune définition de politique d'établissement. Les variables définies dans `toto.config` sont donc traitées sans modification.
- de tester le fichier obtenu, de l'installer à l'emplacement définitif, et de redémarrer `sendmail`.

C'est évidemment le mode d'utilisation adapté aux petits sites (du petit campus jusqu'au particulier utilisant un modem pour accéder à l'Internet.

3.4.2 Définition d'une politique d'établissement

Le deuxième mode d'utilisation est plus évolué : alors que les autres kits définissent un fichier de configuration en fonction de l'utilisateur, le kit de Jussieu définit un fichier de configuration en fonction :

– des variables de l'utilisateur, et
– de la politique de gestion de courriers dans l'établissement.

Autrement dit, il y a deux niveaux de configuration :

1. le niveau de l'administrateur de l'établissement, qui définit la politique. Par exemple :
   – utilisation ou non des MX en interne ;
   – routage centralisé par une machine dédiée au courrier ;
   – noms de machines à 3, 4 ou *n* champs ;
   – espace de noms unique et signature de la forme *Prénom.Nom@domaine.fr*, etc.

   Cette analyse de la politique de l'établissement doit conduire l'administrateur de l'établissement à proposer aux administrateurs des machines des choix très réduits.
2. le niveau de l'administrateur d'une machine, où le choix des options doit être beaucoup plus réduit. Par exemple, au Bureau des Longitudes (`bdl.fr`), le choix est réduit à la question : « *mailhost* ou non ? ».

Cette configuration en deux temps est rendue possible grâce à l'utilisation de variables de très bas niveau dans le configurateur (mais de beaucoup plus haut niveau que la configuration de `sendmail`). Il suffit à l'administrateur de l'établissement de faire un petit *script shell* pour prendre en entrée des variables de l'administrateur de la machine (de très haut niveau), et les convertir dans la syntaxe du Kit. Le configurateur s'occupe du reste.

Tâches de l'administrateur de l'établissement

La tâche de l'administrateur de l'établissement est donc :

1. de réfléchir sur la politique de gestion des courriers au sein de son établissement ;
2. de chercher le kit de configuration de Jussieu en ftp anonyme à l'adresse indiquée en tête de ce chapitre (voir page 61) ;
3. de se familiariser avec les variables de bas niveau du configurateur ;
4. de choisir un jeu d'options pour les administrateurs de son établissement et de réfléchir à leur expression en fonction des variables de bas niveau ;
5. de réaliser un petit *script shell* pour convertir ces options en variable de bas niveau ;
6. de tester ce script ;
7. de l'intégrer au configurateur, à l'endroit spécifié (voir les premières lignes du configurateur) ;
8. de le rendre accessible en ftp anonyme sur son site.

Tâches de l'administrateur de la station

L'administrateur de la station voit sa tâche réduite à sa plus simple expression. Une fois récupéré le configurateur modifié, il change les quelques variables à changer, puis il lance :

```
$ vi ma-station.config
...
$ ./configurateur ma-station.config > sendmail.cf.new
```

et il ne reste plus qu'à tester et installer ce fichier de configuration.

Et s'il n'y a pas d'administrateur d'établissement ?

Dans le cas où il n'y a pas d'administrateur d'établissement, c'est-à-dire lorsque l'établissement est trop petit pour avoir plusieurs administrateurs, il suffit de générer un fichier directement en fonction des variables de bas niveau. Dans ce cas, le script de conversion est vide, le configurateur est appelé directement. C'est le premier mode d'utilisation défini en 3.4.1.

3.5 Présentation des variables

Le but de cette section est de donner une vue synthétique des différentes variables et de leur interaction. Une première partie classe les variables en trois grands groupes. Les parties suivantes mettent l'accent sur quelques points particuliers de la configuration du kit.

La section suivante (section 3.6) constitue la référence exhaustive des variables.

3.5.1 Classification

Les variables peuvent être classées en trois grandes catégories :

1. Configuration de la machine (nom, domaine, emplacement du fichier des aliases, etc.)
   Ces variables sont :

Variable	Signification
`Host`	nom de la machine
`Domaine`	nom du domaine
`ListeDomaines`	noms à reconnaître comme le domaine local (pour les changements de domaine)
`Aliases`	localisation et méthode d'accès à la base d'aliases
`SendmailSt`	localisation du fichier de statistiques
`Mqueue`	localisation de la file d'attente
`TailleMaximum`	taille maximum d'un courrier
`SendmailHf`	localisation du fichier d'aide de `sendmail`
`MailerLocal`	arguments du *mailer* utilisé pour remettre les courriers locaux
`MailerUucp`	arguments du *mailer* utilisé pour la propagation des courriers via UUCP
`SansDNS`	utilise ou non le DNS
`FichierInclude`	fichier à inclure dans le `sendmail.cf` généré

2. Politique de routage de courrier

   C'est là que se posent toutes les questions d'architecture de courrier. Le choix se résume aux questions suivantes :
   - que garder sur le disque de la station ?
   - que redistribuer à l'intérieur de l'entité ?
   - comment faire l'envoi vers l'extérieur de l'entité ?
   - accepte-t-on de relayer les courriers, et plus généralement, fait-on de la lutte anti-*spam* ?
   - y-a-t'il des exceptions à la politique de redistribution ?

   Les variables ci-dessous permettent de définir ces comportements. Pour plus d'explication, voir le paragraphe 3.5.2 ci-après ainsi que la référence de ces variables (section 3.6).

Variable	**Signification**
`AdressesLocales`	type d'adresses reconnues comme étant sur cette machine
`ListeAdressesLocales`	liste d'adresses si la variable précédente indique une liste explicite de machine.
`AdressesInternes`	type d'adresses reconnues comme devant être transmises sur d'autres machines à l'intérieur de notre juridiction
`ListeAdressesInternes`	liste d'adresses si la variable précédente indique une liste explicite de machine.
`MailhostEnInterne`	envoi des courriers « internes » à la machine directement, ou à une machine désignée par le nom `mailhost`
`RelaisExterieur`	les courriers vers l'extérieur doivent être transmis à une machine spécifiée comme étant un relais, ou envoyés directement sur l'Internet en utilisant les MX
`TableRoutages`	liste des exceptions à la politique de redistribution
`SansCanonisation`	suppose que les adresses en entrée sont correctes, et ne fait pas de canonisation (à n'utiliser que dans des cas rares)
`ListeNoire`	activation des règles anti-*spam*, et spécification des domaines *spammeurs* ainsi que des domaines autorisés à nous utiliser comme relais
`CodeErreur`	code d'erreur renvoyé, si les règles anti-*spam* sont activées
`URL`	URL renvoyée avec les messages de rejet dûs aux règles anti-*spam*

3. Réécriture des adresses

   Les champs `From` et assimilés (`Sender`, `Reply-To`, etc.) contiennent des adresses qui peuvent être réécrites. Ceci a deux utilisations fréquemment demandées :
   - faire disparaître le nom de la machine de l'adresse électronique
   - transformer les adresses de type *login*`@...` en *Prénom.Nom*`@...`

   Ces réécritures sont définies par les variables suivantes :

Variable	**Signification**
`RevAliases`	table de correspondance *login* vers *Prénom.Nom*
`ReecritureAdressesLocales`	réécriture de la partie « nom de machine » lorsqu'une adresse est considérée comme étant « locale »
`ReecritureAdressesInternes`	réécriture de la partie « nom de machine » lorsqu'une adresse est considérée comme étant « interne »

3.5.2 Politique de distribution

La figure 3.1 illustre l'impact des variables sur la politique de distribution du courrier électronique. Du point de vue d'une station, on découpe l'espace en trois parties :

– les adresses locales, c'est-à-dire les adresses pour lesquelles les courriers doivent être considérés comme devant rester sur cette station ;
– les adresses internes, c'est-à-dire que cette station se considère comme étant la « tête de pont » d'une entité (un laboratoire, un campus, une équipe, etc.) et se charge de redistribuer le courrier dans cette entité ;
– le reste du monde ;

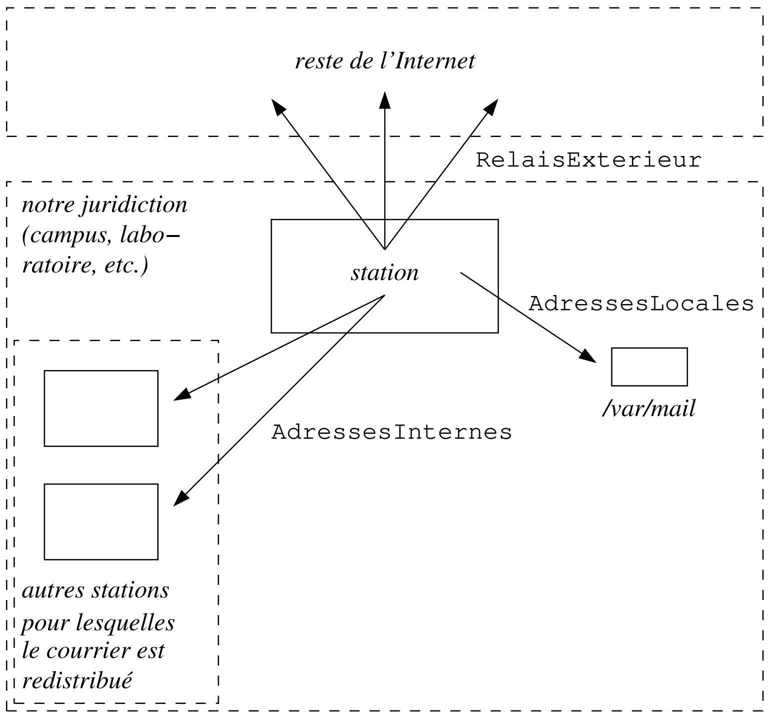

FIG. 3.1 – Politique de distribution du courrier

Pour illustrer ces variables, considérons quelques cas typiques.

Mailhost du campus Jussieu

Sur le campus de Jussieu, une machine particulière (`shiva`) est dédiée à la gestion du courrier (en particulier, tous les MX pointent sur elle). Lorsqu'elle rencontre une adresse du type *toto*`@...`*labo*`.jussieu.fr`, elle l'envoie à une machine particulière (`mailhost.`*labo*`.jussieu.fr`) qui se charge de la politique de redistribution du laboratoire.

La configuration de `shiva.jussieu.fr` est la suivante :

– elle doit reconnaître et garder sur son disque local les courriers adressés à : `shiva.jussieu.fr` et à `jussieu.fr`,

ce qui s'exprime avec la variable :
`AdressesLocales='DOMAINE'`

- elle doit transmettre tous les courriers envoyés à l'intérieur de `jussieu.fr` vers les laboratoires concernés. Autrement dit, la liste des adresses internes est : `*.jussieu.fr` (mais pas `jussieu.fr` seulement puisque c'est déjà reconnu comme une adresse locale), ce qui s'exprime par :
`AdressesInternes='ETOILE_DOMAINE'`
- elle doit transmettre tous les courriers non reconnus comme locaux ou internes sur l'Internet. Dans ce cas, il n'y a pas relais extérieur, ce qui veut dire que les MX sont utilisés pour dialoguer avec l'Internet, et que les courriers UUCP ou BITNET sont envoyés à une passerelle ; ceci s'exprime simplement en mettant à « vide » la variable :
`RelaisExterieur=''`

Laboratoire intégré

Toujours sur le campus de Jussieu, prenons le cas d'un laboratoire `l1` dans lequel l'architecture du courrier est très intégrée : un serveur (le *mailhost*) conserve toutes les boîtes aux lettres sur son disque local. Ceci signifie que tous les courriers adressés à n'importe quelle machine du laboratoire `l1.jussieu.fr` restent sur ce serveur.

Dans ce contexte, la configuration du serveur `mailhost.l1.jussieu.fr` est la suivante :

- il doit reconnaître et conserver sur son disque local les courriers adressés à `l1.jussieu.fr` aussi bien que ceux adressés à n'importe quelle machine (`*.l1.jussieu.fr`), ce qui s'exprime par :
`AdressesLocales='TOUT_DOMAINE'`
- il ne doit distribuer le courrier à aucune machine dans sa juridiction puisqu'il conserve localement tous les courriers du laboratoire :
`AdressesInternes='RIEN'`
- il doit transmettre tous les courriers non reconnus, c'est-à-dire ceux extérieurs au laboratoire (même s'ils sont adressés à l'intérieur de Jussieu) à la machine centrale de Jussieu `mailhost.jussieu.fr` (qui est un simple alias de `shiva`). Ceci s'exprime par :
`RelaisExterieur='smtp.[mailhost.jussieu.fr]'`

Une autre configuration intéressante est la symétrique de la précédente dans le laboratoire `l1.jussieu.fr`. Il s'agit d'une machine quelconque du laboratoire qui ne reçoit pas de courrier[1]. Le répertoire des boîtes aux lettres (`/var/spool/mail`) du serveur est accédé par NFS, ce qui permet à chacun de lire son courrier depuis sa station.

La configuration d'une feuille est donc :

- elle ne doit reconnaître aucun courrier local, car tout doit être envoyé au *mailhost*. La remise physique n'est pas faite par la feuille, mais par le *mailhost*. Donc :
`AdressesLocales='RIEN'`
- elle ne doit redistribuer le courrier à personne puisqu'elle n'est pas responsable d'une entité :
`AdressesInternes='RIEN'`
- tout ce qui n'est pas reconnu jusqu'ici (c'est-à-dire toutes les adresses en fait) doit être envoyé au *mailhost* du laboratoire, ce qui se traduit par :
`RelaisExterieur='smtp.[mailhost.l1.jussieu.fr]'`

Dans cette configuration, une feuille n'a pas besoin de démarrer le démon `sendmail` en mode *serveur* SMTP. Il suffit donc de le lancer, dans le fichier `/etc/rc.local` ou équivalent, par la commande :

```
sendmail -q30m
```

[1] C'est ce que nous appelons une *feuille* par analogie avec la théorie des graphes.

Ces options ont pour effet de ne pas démarrer le serveur SMTP (absence de l'option `-bd`), mais de configurer `sendmail` en mode « démon », avec vérification de la file d'attente toutes les demi-heures. Ce mode et cette vérification sont intéressantes en cas d'arrêt du *mailhost* : cela permet aux feuilles de reprendre automatiquement les courriers envoyés pendant que celui-ci n'était pas disponible.

De la même manière, il n'y a pas besoin de fichier `aliases`. Celui-ci peut être supprimé sans autre forme de procès.

Campus non intégré

Prenons maintenant l'exemple d'un campus non intégré : une station concentre le courrier adressé au domaine (`c.fr`) ainsi qu'à la plupart des stations du domaine. En outre, elle doit redistribuer le courrier à quelques stations citées explicitement. Les noms de machines dans le domaine `c.fr` ont 3 composants.

Dans ce contexte, la configuration de la station `mailhost.c.fr` est :

- elle doit reconnaître comme locales l'adresse `c.fr` ainsi que toutes les adresses dans le domaine (c'est-à-dire toutes les adresses de la forme `*.c.fr`). Ceci est spécifié par :
  `AdressesLocales='TOUT_DOMAINE'`
- elle doit redistribuer le courrier à une liste explicite de machines, contenue par exemple dans le fichier de nom `/usr/local/etc/redistrib`. Ceci est spécifié par les deux variables :
  `AdressesInternes='LISTE'`
  `ListeAdressesInternes='/usr/local/etc/redistrib'`

  Note : on utilise ici une astuce du kit, à savoir qu'une spécification plus précise est prioritaire. Ainsi, la spécification *liste* pour les adresses internes est considérée *avant* la spécification `*.c.fr` pour les adresses locales.
- comme dans le cas de `shiva.jussieu.fr`, les courriers extérieurs sont envoyés directement sur l'Internet :
  `RelaisExterieur=''`

Une station feuille de `mailhost.c.fr` a une configuration très comparable à celle d'une feuille dans l'exemple précédent (architecture de courrier intégrée dans un laboratoire de Jussieu). Elle n'est pas reprise ici.

En revanche, la configuration d'une des stations de la liste est légèrement différente :

- elle doit reconnaître comme locale son adresse uniquement, c'est-à-dire :
  `AdressesLocales='HOST'`
- elle ne doit pas redistribuer le courrier à l'intérieur d'une entité (elle n'est pas responsable d'un groupe de machines) :
  `AdressesInternes='RIEN'`
- toute adresse non reconnue ici est donc soit une autre adresse dans le domaine `c.fr`, soit une adresse dans l'Internet. Deux solutions peuvent être utilisées ici :
  - soit `mailhost.c.fr` est utilisé pour router tous ces courriers :
    `RelaisExterieur='smtp.[mailhost.c.fr]'`
  - soit ces courriers doivent être directement expédiés :
    `RelaisExterieur=''`

Station unique

Le dernier cas intéressant dans notre revue d'exemple consiste en une station isolée, seule station dans un domaine. Dans ce cas, la configuration est particulièrement simple :

– elle doit reconnaître comme locale son adresse uniquement, mais elle peut également reconnaître tout le domaine puisqu'elle est la seule :
`AdressesLocales='TOUT_DOMAINE'`
– il n'y a pas à envisager de redistribution interne dans le domaine :
`AdressesInternes='RIEN'`
– toute adresse non reconnue ici est donc forcément une adresse extérieure, et le message doit être expédié sur l'Internet :
`RelaisExterieur=''`

3.5.3 Centralisation et fiabilisation

De plus en plus d'organisations mettent en place une machine de courrier de site, seule habilitée à émettre et recevoir du courrier de l'extérieur. Cette machine redistribue ensuite les messages en interne. Cette politique est celle, par exemple du campus Jussieu (voir 3.5.2, page 67).

Le passage par une machine unique suppose :

– de placer des MX dans le DNS, pour indiquer au reste de l'Internet que le courrier doit être reçu par cette machine (voir 1.6, page 16) ;
– de mettre en place des filtres sur le routeur d'entrée du site (ou équivalent) pour empêcher les connexions SMTP depuis l'extérieur vers l'intérieur ou réciproquement ;
– d'indiquer aux stations en interne d'utiliser cette machine comme relais (avec la variable `RelaisExterieur`).

Ainsi, la sécurité du site est renforcée. Toutefois, l'inconvénient est la dépendance vis-à-vis d'une machine qui peut tomber en panne, devenir indisponible suite à une coupure de courant, ou toute autre cause.

Pour éliminer cette dépendance, il est possible avec la version 8 de `sendmail` de doubler cette machine. Ainsi :

– les MX doivent indiquer les deux machines, éventuellement avec une priorité équivalente ;
– de modifier les filtres pour autoriser la deuxième machine ;
– d'indiquer aux stations en interne les deux machines comme relais :

```
RelaisExterieur='smtp.[relais1.site.fr]:[relais2.site.fr]'
```

Il est même possible que la machine de secours soit extérieure au site[2]. Ceci peut se faire avec plus ou moins de finesse :

– dans le premier cas, le *mailhost* extérieur ne connaît rien de la politique de routage interne du site secouru. Il dispose alors d'une table de routage simple :

```
site-distant.fr:   smtp.[mailhost.site-distant.fr]
```

[2] C'est par exemple le cas de Jussieu et de l'UVSQ, qui se fournissent mutuellement un service de *mailhost* de secours.

Comme nous l'avons vu en 1.6 (page 16), cela présente surtout l'intérêt d'alléger la reprise, mais pas de fonctionner lorsque le *mailhost* du site distant est indisponible.

– dans le deuxième cas, le *mailhost* extérieur connaît la politique de routage interne du site secouru. Par exemple, l'UVSQ connaît la politique de routage de Jussieu, simplement en énumérant les sous-domaines de Jussieu dans la table de routage[3] :

```
labo1.jussieu.fr:   smtp.[mailhost.labo1.jussieu.fr]
labo2.jussieu.fr:   smtp.[mailhost.labo2.jussieu.fr]
...
```

Pour que cette méthode fonctionne correctement, il faut également qu'il y ait concordance sur la liste noire en cas de lutte anti-*spam* (voir 3.5.4, page 71).

Lorsque le site distant est indisponible, les courriers s'entassent dans la file d'attente du site de secours. Si le site distant reçoit un gros volume de courrier, cela peut devenir génant pour le site de secours : `sendmail` passe alors de plus en plus de temps à explorer sa file d'attente en pure perte – il suffit d'utiliser `mailq` pour s'en convaincre. Si le problème se prolonge, il peut s'avérer utile de migrer la file d'attente dans un autre répertoire :

```
# kill -15 `head -1 /etc/sendmail.pid`      # en version >= 8.8 uniquement
# cd /var/spool
# mv mqueue mqueue.old
# mkdir mqueue
# sendmail -bd -q30m
```

Lorsque le site redevient disponible, il faut vider la file d'attente :

```
# sendmail -oQ/var/spool/mqueue.old -q
```

Attention toutefois : des messages non destinés au site secouru peuvent figurer dans cette file d'attente sauvegardée. Une fois la vieille file d'attente explorée, il restera sans doute des messages. Dans ce cas, il faut les déplacer dans la file d'attente « normale » (un simple `mv` suffit) après s'être assuré de ne pas écraser de message dans cette file « normale ».

3.5.4 Lutte anti-spam

Dans le cadre de la lutte anti-*spam* (voir 1.10, page 34), le kit intègre dorénavant des ensembles de règles (voir 2.5.5, page 48) destinés à se protéger contre une partie des *spams*. Bien sûr, les règles ne peuvent pas être efficaces à 100 %, mais nous avons constaté une nette diminution en pratique.

Lutte anti-spam derrière un relais de messagerie

Si la machine a configurer est située derrière un relais de messagerie qui n'a pas de règles anti-spam, et en particulier pas de liste noire, il est possible de mettre en place des règles particulières. Voir le paragraphe 3.7.4, page 116 pour un exemple de filtrage sur le champ `Received`.

[3]Cette énumération est réalisée automatiquement par un script *shell* qui explore le DNS.

Le reste de cette section décrit la lutte anti-spam appliquée sur un relais de messagerie.

Actions effectuées

Les règles implémentent les actions suivantes :

– refus des clients SMTP qui ne sont pas enregistrés dans le DNS (optionnel, voir page 74) ;
– refus de servir de relais pour des courriers qui ne nous concernent pas ;
– refus des courriers avec une adresse d'expéditeur invalide ;
– refus des courriers émanant d'une adresse (de client SMTP ou d'expéditeur) enregistrée dans la liste noire locale ;
– refus des courriers émanant d'un client SMTP enregistré dans une des listes noires de type RBL (voir 1.10.2, page 36). Ce test est optionnel (voir page 73).

Ces actions sont activées, très simplement, lorsque la variable `ListeNoire` est définie. En plus, cette variable définit l'accès à un fichier, dont le contenu est décrit ci-après.

Filtrage au niveau du routeur

Pour implémenter une politique de lutte anti-*spam* efficace, il est très fortement conseillé d'adopter une démarche centralisée, avec une seule[4] machine accessible de l'extérieur. Seule cette machine mettra en œuvre les règles décrites ici. Les configurations des autres machines du site n'auront pas besoin de prendre en compte la lutte anti-*spam*.

Cette politique suppose un filtrage sur le port SMTP (25) sur le routeur d'entrée du site, ou toute autre méthode équivalente.

Liste noire

Le fichier spécifié par la variable `ListeNoire` contient les adresses à refuser, ainsi que les adresses extérieures pour lesquelles nous servons de relais. Ce fichier est constitué de lignes ayant la syntaxe suivante :

objet	*mot-clef*

Avec :

– le champ *objet*, composé de :
  – un domaine (exemple : `cyberpromo.com`)
  – une adresse IP complète (exemple : `204.137.223.1`) ou incomplète (exemple : `204.137.223`)
  – un utilisateur (exemple : `mechant@aol.com`)
  – un code activant une caractéristique particulière :
    – `NUMERIQUE` : pour accepter les clients SMTP non enregistrés dans le DNS ;
    – `RBL` : pour tester si le client SMTP est enregistré dans une des bases de données de type RBL (*Realtime Blackhole List*).

[4]Ou éventuellement deux, pour assurer la tolérance aux pannes.

- le champ *mot-clef*, contenant un des trois mot-clefs ci-après :
  - `SPAM` : si le courrier doit être rejeté
  - `OK` : si on accepte le courrier (ce qui peut permettre de faire des exceptions fines dans des domaines *spammeurs*)
  - `LOCAL` : pour un domaine ou une adresse pour le compte desquels on accepte de relayer des messages
- le code `RBL` est un cas particulier, pour lequel `SPAM` doit être suivi d'une liste de domaines séparés par des caractères « : ».

Par exemple :

```
# utiliser deux listes noires
RBL                         SPAM relays.ordb.org:inputs.orbz.org
# liste des domaines spammeurs
cyberpromo.com              SPAM
gentil.cyberpromo.com       OK
204.137.223                 SPAM
mechant@aol.com             SPAM
# les sites pour lesquels on accepte de relayer
domaine.fr                  LOCAL
eudora.baladeur.edu         LOCAL
```

Ceci provoquera le rejet de tout courrier provenant de `cyberpromo.com`, sauf s'il s'agit de la machine `gentil` de ce domaine. L'ordre des lignes n'a pas d'importance, les règles du kit cherchent toujours du plus précis vers le moins précis. De même, tout courrier signé de l'utilisateur `mechant@aol.com`, ou venant de l'une des adresses du réseau 204.137.223 sera rejeté.

En revanche, nous acceptons de servir de relais pour le domaine `domaine.fr`, ainsi que pour la machine isolée `eudora.baladeur.edu` qui peut être l'adresse d'un utilisateur local, expatrié temporairement sur un autre site, mais qui continue à traiter son courrier (via POP et SMTP) à distance.

Code et message d'erreur

Comme décrit en 2.5.5 (page 49), le code d'erreur renvoyé par les règles anti-*spam* est important. La variable `CodeErreur` permet de changer la valeur par défaut. Si le besoin s'en fait sentir, deux codes d'erreurs peuvent être spécifiés, afin de distinguer suivant la cause. Pour plus de détails, voir la variable `CodeErreur` (3.6.12, page 89).

En outre, l'expérience montre que les règles anti-*spam* affectent également des sites honnêtes, mais mal configurés. Afin de leur permettre de corriger leur configuration et d'expliquer aux utilisateurs pourquoi leur courrier a été rejeté, les messages incluent l'URL d'une page Web[5]. Vous êtes encouragés à récupérer cette page, l'adapter éventuellement à votre site, et la mettre à disposition sur votre propre serveur Web.

Utilisation de listes de type RBL

Si la ligne :

```
RBL        SPAM domaine₁ : domaine₂ : ...
```

[5] http://www.prism.uvsq.fr/~pda/kit-jussieu/rejet

est présente dans le fichier spécifié par la variable `ListeNoire`, `sendmail` active la recherche de l'adresse du client SMTP dans les arborescences DNS $domaine_i$ (voir 1.10.2, page 36). Si elle est trouvée dans au moins un des domaines, la connexion est rejetée.

Clients SMTP non enregistrés dans le DNS

Les connexions sont refusées par les règles anti-*spam* si le client SMTP n'est pas correctement enregistré dans le DNS[6]. Ceci se traduit, pour l'expéditeur, par le renvoi du courrier avec le message « *Unknown relay name* ».

Hormis les cas manifestes où le client SMTP n'est pas du tout enregistré dans le DNS, ce genre de refus peut également arriver si le client n'est qu'*à moitié* enregistré dans le DNS, c'est-à-dire dans la zone normale, mais pas dans la zone inverse. On peut le vérifier avec la commande `nslookup` :

```
$ nslookup
> machin.bidule.fr.
...
Name:     machin.bidule.fr
Address:  1.2.3.4
> set q=ptr
> 4.3.2.1.in-addr.arpa.
...
*** can't find 4.3.2.1.in-addr.arpa.: Non-existent host/domain
```

La première interrogation (`machin.bidule.fr`) donne bien l'adresse IP correspondant au nom, mais la deuxième interrogation (dans le domaine `in-addr.arpa`) montre que l'adresse IP ne peut pas être convertie en nom, donc il manque l'enregistrement de type PTR dans l'arborescence `in-addr.arpa`.

Le refus de ces adresses présente des avantages et des inconvénients. Parmi les avantages, on peut citer :

- le rejet de quelques *spams* supplémentaires ;
- le « nettoyage » du réseau : normalement, les règles de bonne conduite sur Internet font que toute machine devrait avoir une adresse IP correctement enregistrée. De plus en plus de serveurs `ftp` anonyme, par exemple, imposent cette contrainte.

Parmi les inconvénients, on peut citer :

- une petite charge supplémentaire pour l'administrateur de la messagerie, qui doit parfois chercher pourquoi certains utilisateurs ne peuvent plus recevoir de messages en provenance de leurs correspondants ;
- le rejet abusif de certains messages : cela se produit par exemple lorsque le client est à l'autre bout de la planète, que ses serveurs DNS sont lents à répondre, et que le code d'erreur ne permet pas au client d'essayer à nouveau.

Si les inconvénients l'emportent sur les avantages, il est possible d'annuler cette contrainte et d'accepter les clients SMTP non enregistrés dans le DNS. Il suffit pour cela d'ajouter la ligne suivante dans le fichier spécifié par la variable `ListeNoire` :

```
NUMERIQUE        OK
```

[6] Ce comportement n'est théoriquement pas « légal », si on se réfère aux RFC. Toutefois, l'expérience montre que de nombreux *spams* sont arrêtés avec cette méthode.

Si pour une raison ou une autre, des clients SMTP locaux au site ne peuvent pas être enregistrés dans le DNS, il est toutefois possible de les autoriser à relayer, avec une ligne ressemblant à celle-ci :

```
193.51.24       LOCAL
```

Analyse des logs

À chaque fois qu'un courrier est rejeté, `sendmail` le signale dans le fichier de *log*. Le format exact dépend de la version de `sendmail`, mais cela donne par exemple :

```
Feb  1 00:17:55 soleil sendmail[4254]: AAA04254:
     ruleset=check_mail,
     arg1=<TryToRecycle@Conserve.net>,
     relay=smarttalk.com [192.41.47.56],
     reject=451 <TryToRecycle@Conserve.net>... Sender address must resolve
```

Détaillons cet exemple[7], découpé pour la lisibilité en plusieurs lignes :

- la première ligne, comme d'habitude, donne des informations sur la date, l'heure, la machine, le programme et l'identificateur dans la file d'attente de `sendmail` ;
- la deuxième ligne indique que l'action est une erreur détectée dans la règle `check_mail` (donc lors de la réception de l'adresse d'émetteur, voir 2.5.5, page 49) ;
- la troisième ligne fournit le premier argument de la règle, c'est-à-dire l'adresse du récepteur figurant dans l'enveloppe
- la quatrième ligne mentionne le client SMTP (nom et adresse IP) ;
- la cinquième et dernière ligne donne la raison de l'erreur (code et message en clair) : ici, l'adresse fournie (`TryToRecycle@Conserve.net`) ne correspond à aucune adresse réelle (la partie après le « `@` » n'existe pas dans le DNS).

Ce message est un excellent exemple de *spam* : une adresse inexistante, et un relais qui n'a aucun rapport avec l'adresse spécifiée par le *spammeur*.

Les différentes causes d'erreur que l'on peut voir dans les logs sont les suivantes[8] :

- `Go away`
  L'adresse ou le nom du client SMTP, ou de l'expéditeur, figure dans la liste noire locale.
- `Unknown relay name`
  Le client SMTP n'est pas correctement enregistré dans le DNS.
- `Rejected address by xxxxx`
  L'adresse du client SMTP figure dans la liste noire *xxxxx* (de type RBL).
- `Polite people give a qualified address`
  L'adresse d'expéditeur n'est pas correctement qualifiée : c'est typiquement le cas d'un *spammeur* qui essaye de nous faire croire que son adresse est locale, pour mieux nous utiliser comme relais.

[7]Pris au hasard dans 10 000 rejets d'un dimanche ordinaire...
[8]Ces messages sont extraits du configurateur à la date de l'écriture de cette documentation. Ils peuvent ne plus être en phase avec les versions ultérieures du configurateur.

- `Sender address must resolve`
  L'adresse d'expéditeur n'existe pas. Les deux cas les plus typiques sont les *spammeurs* qui mettent une adresse farfelue, ou les utilisateurs qui configurent mal leur courrier : adresse avec une faute d'orthographe, mal qualifiée, etc. De toutes manières, leurs correspondants ne pourront pas faire de *reply*.
- `Relaying denied`
  Un client SMTP externe essaye de nous envoyer un courrier destiné à une adresse externe, en signant avec une adresse d'expéditeur interne. C'est typiquement le cas où on essaye de nous utiliser comme relais. Il est éventuellement possible d'autoriser cela, si c'est justifié, en ajoutant une ligne `OK` au fichier spécifié par la variable `ListeNoire`.

Le programme `stat-spam`, fourni dans le même répertoire que le kit de Jussieu, permet d'exploiter le fichier de *log* de `sendmail` (version $\geq$ 8.8.8) et d'en extraire des statistiques.

3.5.5 Accès aux fichiers, aux NIS et à LDAP

Le fichier `sendmail.cf` généré utilise un certain nombre d'informations auxiliaires, issues de fichiers ou de bases LDAP. Le tableau ci-dessous récapitule ces informations :

Variable	Description	Compilé	LDAP	NIS
`Aliases`	aliases	oui	oui	oui
`RevAliases`	correspondance *login* vers *Prénom.Nom*	oui	oui	oui
`TableRoutages`	exceptions de la politique de routage	oui	oui	non
`ListeNoire`	adresses filtrées, acceptées ou relayées	oui	oui	non
`ListeAdressesLocales`	liste explicite des adresses locales	non	non	non
`ListeAdressesInternes`	liste explicite des adresses internes	non	non	non

Dans ce tableau :

- la première colonne cite la variable référençant l'information ;
- la deuxième colonne rappelle sa signification ;
- la troisième colonne indique si l'information peut être issue d'un fichier « compilé » ou non ;
- la quatrième colonne indique si l'information peut être obtenue à partir d'une base LDAP ou non ;
- la cinquième colonne indique si l'information peut être obtenue à partir d'une *map* NIS ou non ;

Accès via des fichiers compilés

Dans la plupart des cas, les informations sont contenues dans les fichiers. Ceux-ci doivent être « compilés » pour maintenir de bonnes performances. La compilation consiste à produire un ou plusieurs fichiers binaires, accélérant ainsi l'accès à une valeur à partir d'une clef. Il y a deux formats de fichiers binaires :

- le format *dbm* : il se traduit par les deux fichiers binaires suffixés par `.dir` et `.pag`. C'est la méthode originelle, mais qui n'est plus guère utilisée aujourd'hui ;
- le format *db* : plus moderne, utilisé seulement par la version 8 de `sendmail`, il est portable entre architectures, plus rapide, plus compact, etc. C'est le format de choix à présent. Le fichier compilé est suffixé par `.db`.

Ce format comprend deux méthodes de compilation : la méthode *hash* (basée sur une fonction de hachage) et la méthode *btree* basée sur un arbre binaire de recherche. Nous utilisons de préférence la méthode *hash*, correspondant à la méthode par défaut pour le fichier des aliases.

Accès via une base LDAP

Si on le souhaite, on peut également stocker les informations dans une base LDAP, ce qui peut simplifier les opérations de centralisation, de maintenance, et de distribution.

Il est possible d'utiliser n'importe quel schéma, puisque la définition des variables inclut le filtre de recherche. Toutefois, les auteurs du kit conseillent d'utiliser le schéma par défaut de `sendmail` (voir 2.6, page 50) pour suivre dès le départ une tentative de standardisation.

La variable `ParametresLDAP` permet de préciser des valeurs par défaut pour toutes les requêtes LDAP, comme l'adresse du serveur, son numéro de port, et le DN de base.

Accès via les NIS

Deux fichiers peuvent être remplacés par des *maps* NIS : le fichier des aliases et le fichier des revaliases. L'accès comme une *map* NIS n'est intéressant que si la station à configurer n'est pas le serveur NIS maître.

L'utilisation des *aliases* est prévue en standard dans les NIS. En revanche, l'utilisation du fichier des *revaliases* nécessite la création d'une nouvelle *map* NIS.

Un dernier point nécessite un avertissement : une *map* NIS est toujours lue à partir d'un fichier compilé en format `dbm`. La version 8 compile par défaut les aliases en format `db`. Il faut en tenir compte.

3.5.6 Gestion de domaines virtuels

Le kit permet de gérer des *domaines virtuels*. Il s'agit de domaines ne correspondant à aucune machine réelle. Deux exemples illustrent particulièrement bien cette situation :

- le laboratoire PRiSM à l'Université de Versailles s'appelait auparavant le laboratoire MASI.
  La machine `mailhost.prism.uvsq.fr` reçoit donc à la fois du courrier pour `prism.uvsq.fr` (le domaine réel) et pour `masi.uvsq.fr` (le domaine virtuel).
- toujours dans cette université, une machine (`lune.uvsq.fr`) est dédiée au courrier des laboratoires ou services qui ne peuvent investir dans un *mailhost* et son administration (typiquement des services ou laboratoires ayant des réseaux de micros).
  Cette machine dédiée accueille des comptes courrier (avec POP et IMAP, voir 1.9, page 32). Chaque utilisateur appartient à un laboratoire ou service *x*. En conséquence, son adresse est *utilisateur*`@`*x*`.uvsq.fr` bien que le domaine *x*`.uvsq.fr` ne corresponde à aucune machine réelle.

Il y a trois manières *exclusives* de gérer ces domaines :

1. reconnaissance simple : toute adresse appartenant au domaine virtuel est réécrite en une adresse dans le domaine réel (exemple du laboratoire PRiSM ci-dessus) ;

2. reconnaissance locale sur le nom d'utilisateur : suivant une base d'aliases, certains utilisateurs ont une adresse dans un domaine virtuel (exemple de `lune.uvsq.fr` ci-dessus) ;
3. reconnaissance distante sur le nom d'utilisateur : cette méthode est équivalente à la précédente, sauf que la reconnaissance et la traduction d'adresse (donc la base d'aliases) sont effectuées sur une machine distante (typiquement le *mailhost* du site).

Les paragraphes suivants détaillent ces trois méthodes. **Attention : il ne faut en sélectionner qu'une seule sur les trois !** Toute tentative de modifier à la fois `ListeDomaines` *et* `TableRoutages` se soldera par un échec ! Toutefois, quelle que soit la méthode choisie, il faut penser à ajouter un RR de type MX dans le DNS (voir paragraphe 1.6, page 16) vers la machine sur laquelle doit arriver le courrier pour le domaine virtuel.

Reconnaissance simple

Cette première méthode est déconseillée, bien qu'elle soit particulièrement simple : il suffit d'initialiser la variable `ListeDomaines` avec la liste des domaines à reconnaître. Par exemple, la machine `mailhost.prism.uvsq.fr` doit reconnaître également l'ancien domaine `masi.uvsq.fr`). Il suffit donc d'initialiser cette variable :

```
ListeDomaines='prism.uvsq.fr prism masi.uvsq.fr masi'
```

Attention toutefois : tout courrier entrant ou sortant avec une adresse dans le domaine virtuel est réécrit en utilisant le domaine réel. Ainsi, lorsque le courrier suivant est traité, les adresses sont réécrites comme suit :

```
From: pda@masi.uvsq.fr                    From: pda@prism.uvsq.fr
To: dugenoux@masi.uvsq.fr        ==>      To: dugenoux@prism.uvsq.fr
Subject: essai                            Subject: essai
```

Comme cette transformation s'applique sur le domaine (`masi.uvsq.fr`) et non sur le nom d'utilisateur, il n'y a pas besoin de base d'aliases.

En pratique, ce cas est trop simpliste : il ne peut convenir qu'à un changement de domaine (renommage d'une organisation, avec maintien des anciennes adresses un certain temps). De plus, il présente un désavantage majeur puisque les adresses sont réécrites, et qu'il sera impossible de savoir si l'ancien domaine est encore utilisé. Comme elle donne rarement les résultats attendus, et qu'elle a des effets de bords peu souhaitables, on lui préférera l'une des deux méthodes suivantes.

Reconnaissance locale sur le nom d'utilisateur

Dans cette configuration, le traitement est effectué sur la machine devant émettre et recevoir les courriers (typiquement le *mailhost*). Normalement, un utilisateur a une adresse de la forme ...`@`*r* où *r* est le domaine réel. Il faut que pour un (ou plusieurs bien sûr) utilisateur *u*, l'adresse soit réécrite en *u*`@`*v* où *v* est le domaine virtuel.

Prenons un exemple[9] : le CCR (Centre de Calcul Recherche) de Jussieu abrite le compte du Président de l'Université Pierre et Marie Curie (UPMC). Les adresses électroniques des utilisateurs sont normalement de la forme ...`@ccr.jussieu.fr`, mais le compte `toto` doit avoir son adresse réécrite en `President@upmc.jussieu.fr`.

[9]Fictif à l'heure où nous écrivons ces lignes.

Cette modification doit être faite au CCR, et bien entendu, les courriers envoyés à cette adresse doivent parvenir à l'utilisateur `toto`.

Pour ce faire, il suffit de configurer le DNS, et plus précisément la zone `jussieu.fr`, comme suit :

```
; zone jussieu.fr
upmc            IN      MX      100 shiva.jussieu.fr.
mailhost.upmc   IN      CNAME   moka.ccr.jussieu.fr.
```

La ligne `upmc` définit un MX pour `upmc.jussieu.fr`. Comme n'importe quelle autre entité de Jussieu, il faut avertir le monde entier que c'est le *mailhost* de Jussieu (c'est-à-dire `shiva`) qui doit recevoir le courrier.

La ligne suivante (`mailhost.upmc`) permet de ne rien modifier sur `shiva` pour traiter ces adresses `upmc` : comme pour n'importe quelle autre entité de Jussieu, `shiva` doit envoyer les courriers pour `upmc.jussieu.fr` à la machine `mailhost.upmc.jussieu.fr`, c'est-à-dire au *mailhost*[10] du CCR (`moka.ccr.jussieu.fr`).

C'est sur `moka`, le *mailhost* du CCR, que doivent être concentrées toutes les modifications. Il suffit d'initialiser les variables suivantes.

- Pour la réception des messages :
  - `TableRoutages`
    Cette variable doit pointer sur un fichier qui doit contenir en tout et pour tout :
    ```
    upmc.jussieu.fr:   local.NIMPORTEQUOI
    ```
    Cette ligne indique à `moka` que tout courrier reçu adressé à `upmc.jussieu.fr` doit être envoyé vers l'agent de remise (*mailer*) `local`. En fait, ceci provoque une lecture du fichier des aliases. La chaîne après le point (ici `NIMPORTEQUOI`) doit contenir quelque chose (chaîne de caractères non vide), même si le contenu n'a pas d'importance.
  - `Aliases`
    Le fichier des aliases doit contenir en particulier une ligne pour chaque adresse dans le domaine virtuel :
    ```
    President@upmc.jussieu.fr:   toto
    ```
    Contre toute apparence, cette adresse (`President@upmc.jussieu.fr`) est bien une adresse locale, comme spécifié dans la table de routages. La compilation du fichier des aliases avec `newaliases` ne pose donc pas de problème.
- Pour l'émission des messages :
  - `RevAliases`
    Cette variable doit pointer sur un fichier. Celui-ci doit contenir, en plus des lignes habituelles :
    ```
    toto:   President@upmc.jussieu.fr
    ```

Attention : il ne faut pas oublier de *compiler* ces trois fichiers (voir paragraphe 3.5.5, page 76).

La gestion de plus d'un domaine virtuel ne pose pas de difficulté particulière : il suffit d'ajouter la ligne correspondante dans la table des routages, ainsi qu'une ligne par utilisateur dans les fichiers d'aliases et d'aliases inverses.

[10] On rappelle qu'un MX ou un CNAME doit pointer sur un nom canonique, pas sur un alias.

Reconnaissance distante sur le nom d'utilisateur

Il s'agit ici de concentrer tout le travail de reconnaissance et de réécriture sur une machine différente de celle qui héberge les comptes des utilisateurs du domaine *virtuel*.

Pour cela, reprenons le cas précédent et supposons, cette fois-ci, que l'on désire réaliser la reconnaissance sur le *mailhost* général de l'Université (`shiva.jussieu.fr`) et non sur le *mailhost* du CCR. Cela pourrait s'avérer intéressant dans le cas où on préférerait une administration centralisée.

Dans ce cas, le *mailhost* du CCR n'a besoin d'aucune modification.

La configuration du DNS est simplifiée :

```
; zone jussieu.fr
upmc              IN        MX        100 shiva.jussieu.fr.
```

Ici, il n'y a plus besoin de `mailhost.upmc` puisque nous allons voir que le routage s'effectue explicitement et non plus via le DNS. Cette modification du DNS n'est donc plus utilisée que pour annoncer l'existence du domaine virtuel `upmc.jussieu.fr`.

Le *mailhost* de `jussieu.fr` doit bien sûr être modifié. Voici les nouvelles variables de configuration :

- Pour la réception des messages :
  - `TableRoutages`
    Cette variable doit pointer sur un fichier qui doit contenir en tout et pour tout :
    ```
    upmc.jussieu.fr:   local.NIMPORTEQUOI
    ```
    Cette ligne indique à `shiva` que tout courrier reçu adressé à `upmc.jussieu.fr` doit être envoyé vers l'agent de remise (*mailer*) `local`. Comme précédemment, ceci provoque une lecture du fichier des aliases.
  - `Aliases`
    Le fichier des aliases doit contenir en particulier une ligne pour chaque adresse dans le domaine virtuel :
    ```
    President@upmc.jussieu.fr:   toto@ccr.jussieu.fr
    ```
    Comme précédemment, cette adresse est locale et le fichier des aliases peut donc être compilé avec la commande `newaliases`.
- Pour l'émission des messages :
  - `RevAliases`
    Cette variable doit pointer sur un fichier. Celui-ci doit contenir une ligne pour chaque adresse distante à transformer en adresse virtuelle :
    ```
    toto@ccr.jussieu.fr   President@upmc.jussieu.fr
    ```
    Attention : l'adresse figurant en partie gauche n'est plus une adresse locale. Le fichier ne peut donc plus être compilé avec l'astuce « `sendmail -bi` ». Il faut le compiler avec `makemap` (voir le paragraphe 3.6.16, page 92).

3.5.7 Particularités propres à un site

Le domaine de la messagerie est par nature un monde où l'homogénéïté ne règne pas toujours en maître. Il est donc inévitable que certains sites, utilisant ce kit, développent leurs propres extensions. À titre d'exemple, le laboratoire

PRiSM à l'université de Versailles doit gérer plusieurs types d'adresses anciennes (en plus de `masi.uvsq.fr` décrit précédemment), dont certaines sont toujours valides sur le campus de Jussieu. Cette situation ne peut pas être traitée par un outil très général, et il faut modifier le fichier `sendmail.cf` « à la main ».

Pour ne pas perdre les modifications propres à un site, et pour pouvoir les réintégrer dans les nouvelles versions du kit ou de fichiers de configuration, deux méthodes peuvent être utilisées.

Inclure un fichier

Il est possible d'inclure un fichier dans le `sendmail.cf` généré par le kit, par le biais de la variable `FichierInclude`. Celle-ci indique un fichier, qui doit exister lors de la génération, dont le contenu est inséré juste avant le premier ensemble de règles.

Il est ainsi possible d'ajouter des définitions (mailers, options, macros, règles, etc.), mais également de modifier des définitions du kit car les nouvelles valeurs remplacent les anciennes.

Cette possibilité convient bien pour des modifications qui n'affectent pas le fonctionnement des règles de réécriture, comme par exemple :

- on désire parfois que les courriers restent dans la file d'attente plus longtemps avant d'être rejetés. Par exemple, les travaux de désamiantage du campus Jussieu imposent parfois d'arrêter les machines d'un laboratoire pendant quelques jours. Comme le temps par défaut du kit est de 3 jours, le *mailhost* de Jussieu inclut un fichier contenant :

```
_#
_# Pendant les travaux de désamiantage...
_#
OT8d
```

- certains sites préfèrent modifier le comportement par défaut de `sendmail` vis-à-vis des caractères 8 bits. Ainsi, ces sites peuvent inclure le fichier contenant :

```
_#
_# Toujours communiquer les 8 bits, même si ça ne respecte pas les standards...
_#
O EightBitMode=pass
```

Attention : les commentaires, dans ce fichier, commencent par les deux caractères « `_#` ».

Faire un patch

L'autre méthode consiste à faire un *patch*[11].

Par exemple, supposons que le fichier généré par le configurateur s'appelle `sendmail.cf.org`, et que le fichier `sendmail.cf` soit dérivé de ce fichier en intégrant des modifications locales. La commande :

[11] Si vous ne disposez pas de l'utilitaire `patch`, toute bonne archive GNU, telle que `ftp.jussieu.fr:/pub/gnu devrait pouvoir vous le fournir.`

```
diff -C 20 sendmail.cf.org sendmail.cf > patch.kit
```

crée un fichier de différences nommé `patch.kit` contenant les modifications locales. Par la suite, pour appliquer ces différences sur un fichier nouvellement généré par le kit, il suffira d'appeler la commande `patch` :

```
patch < patch.kit
```

Stéphane Bortzmeyer propose également d'utiliser un fichier `Makefile` pour automatiser la confection du *patch*, puis son utilisation. Le fichier `Makefile` ci-dessous constitue un bon exemple :

```
REGLES = ./regles
CONFIG = ./machine.config
PATCH  = patch.kit
CF     = sendmail.cf

all:
    @echo "Que voulez-vous faire ?"
    @echo "    make diff        : genere un patch"
    @echo "    make sendmail.cf : genere un fichier sendmail.cf"

diff:
    ./configurateur $(REGLES) $(CONFIG) > sendmail.cf
    cp sendmail.cf sendmail.cf.OLD
    cp /etc/sendmail.cf sendmail.cf
    -diff -c sendmail.cf.OLD sendmail.cf > $(PATCH)

sendmail.cf: $(REGLES) $(CONFIG) $(PATCH)
    ./configurateur $(REGLES) $(CONFIG) > sendmail.cf
    patch < $(PATCH)
```

La méthodologie est alors la suivante :

1. l'administrateur modifie le fichier `sendmail.cf` et l'adapte à son site ;
2. lorsque les modifications sont effectuées, il suffit de faire :

   ```
   make diff
   ```

   pour générer un fichier de différences ;
3. par la suite, si une nouvelle version du configurateur sort, ou si le fichier de variables est modifié, il suffit de faire :

   ```
   make sendmail.cf
   ```

   pour générer le fichier `sendmail.cf` et appliquer le patch qui réintègrera les modifications locales.

Les deux méthodes

Bien sûr, rien ne vous empêche d'utiliser ces deux méthodes simultanément. Par exemple, il peut être souhaitable de minimiser les modifications par *patch*, et de placer le maximum de modifications dans le fichier inclus.

3.6 Description détaillée des variables

Le configurateur est entièrement paramétré à l'aide des variables décrites dans cette partie. Le type, la signification et au moins un exemple sont donnés pour chaque variable.

Le kit utilisant le préprocesseur du langage C (`cpp`) pour sélectionner les options ou indiquer les paramètres, il faut tenir compte des particularités propres à chaque processeur. Par exemple, si votre machine s'appelle de manière très originale `sun` et que vous lancez le configurateur sur un Sun, le préprocesseur remplace le nom de votre machine par `1`, ce qui va semer la confusion... La page de manuel de `cpp` indique généralement les symboles prédéfinis.

3.6.1 Host

- type : chaîne
- signification : cette variable contient le nom de la station, c'est-à-dire la partie la plus à gauche du FQDN (*Fully Qualified Domain Name*)
- exemple : pour `cezanne.prism.uvsq.fr`, cette variable contient `cezanne`.
- remarques :
  - ce nom ne doit pas contenir de point,
  - ce nom ne peut pas être obtenu mécaniquement car il obligerait le configurateur à être exécuté sur la station à configurer, ce qui n'est pas nécessaire.
  - dans certaines versions de `sendmail`, la macro `w` contient le nom de la station. Toutefois, elle ne contient pas toujours le nom non qualifié, cela dépend de la version utilisée. On ne peut donc pas faire confiance à cette macro.

3.6.2 Domaine

- type : chaîne
- signification : nom complet du domaine courant.
- exemple : pour `cezanne.prism.uvsq.fr`, cette variable contient `prism.uvsq.fr`.
- remarque : un nom de domaine sans point est illégal.

3.6.3 ListeDomaines

- type : liste
- signification : noms à reconnaître comme étant le domaine courant.
- exemples :
  - pour `mailhost.lptl.jussieu.fr`, cette variable doit contenir `lptl.jussieu.fr`, et éventuellement `lptl`, de façon à recevoir les courriers destinés à *x*`@lptl` ou *x*`@lptl.jussieu.fr` ;
  - pour `mailhost.prism.uvsq.fr` qui doit reconnaître `prism.uvsq.fr` et `masi.uvsq.fr`, cette variable doit être initialisée à :
    `prism.uvsq.fr`, `prism`, `masi.uvsq.fr` et `masi`.
- remarques :
  - le paragraphe 3.5.6 (page 78) décrit l'utilisation de cette variable pour reconnaître un domaine virtuel.

3.6.4 AdressesLocales

– type : mot-clef RIEN, HOST, DOMAINE, TOUT_DOMAINE, LISTE ou LISTE_DOMAINE
– signification : spécifie quelles sont les adresses à reconnaître comme locales, c'est-à-dire pour lesquelles les courriers restent sur cette machine.
  Ceci permet de centraliser (ou non) le courrier sur un serveur (le *mailhost*) et de permettre à toutes les autres machines de la « zone d'influence » (les *feuilles*) de monter le répertoire des boîtes aux lettres par NFS, partageant ainsi de manière transparente le courrier. Dans le cas d'une centralisation de ce type, les feuilles doivent tout envoyer (même les courriers locaux) au *mailhost* qui se charge de la remise physique.
  La signification des mot-clefs est donnée par le tableau :

Mot-clef	**Signification**
`RIEN`	aucun courrier ne doit rester sur cette machine
`HOST`	seuls les courriers destinés à `host.domaine` doivent rester sur cette machine
`DOMAINE`	seuls les courriers destinés à `host.domaine` et à `domaine` doivent rester sur cette machine
`TOUT_DOMAINE`	tous les courriers adressés à `*.domaine` et à `domaine` doivent rester sur cette machine
`LISTE`	seuls les courriers adressés à `host.domaine` ou aux machines d'une certaine liste doivent rester sur cette machine
`LISTE_DOMAINE`	seuls les courriers adressés à `host.domaine`, `domaine` ou aux machines d'une certaine liste doivent rester sur cette machine

– exemples :
  – `DOMAINE` pour le *mailhost* de Jussieu, qui reçoit seulement les courriers destinés à `jussieu.fr` ou à `shiva.jussieu.fr`,
  – `TOUT_DOMAINE` pour le serveur d'un laboratoire dans lequel tout le courrier est entreposé sur ce serveur,
  – `RIEN` pour le cas symétrique d'une station du même laboratoire qui envoie tout le courrier au serveur. Dans ce cas, cette station n'a pas son propre répertoire des boîtes aux lettres, elle monte celui du serveur par NFS,
  – `HOST` pour une station qui reçoit son courrier et uniquement son courrier,
  – `LISTE` pour un *mailhost* d'un domaine, qui garde en local le courrier de quelques stations et qui redistribue le courrier des autres stations,
  – `LISTE_DOMAINE` pour le serveur d'un laboratoire dans lequel une partie seulement des courriers doit être entreposée sur le serveur, et l'autre partie est redistribuée à quelques autres stations ;
– remarques :
  – cette variable interagit fortement avec `AdressesInternes`. Toutes les combinaisons ne sont pas significatives, voir la description de cette autre variable.
  – si cette variable contient `LISTE`, la variable `ListeAdressesLocales` doit contenir la liste des stations pour lesquelles cette station conserve le courrier.
  – le cas `RIEN` est assez spécial : il correspond au cas d'une station qui utilise le répertoire des boîtes aux lettres via NFS par exemple. Dans cette configuration, il n'y a pas besoin de lancer `sendmail` avec l'option `-bd` (la station ne doit pas recevoir de courrier), et le fichier `aliases` peut être supprimé puisque l'agent de remise physique n'est jamais appelé.

3.6.5 ListeAdressesLocales

– type : liste de noms de machines, ou nom de fichier (si le premier caractère est un « `/` »).

- signification : si la variable `AdressesLocales` = `LISTE` ou `LISTE_DOMAINE`, cette variable contient la liste explicite des machines pour lesquelles le courrier doit être conservé localement.
- exemples :
  - `moka baba chou`
    Ici, une liste de machines est définie.
  - `/usr/local/etc/mail/local`
    Dans ce cas, puisque le premier caractère est un « `/` », c'est un fichier qui contient les noms de machines.
- remarques :
  - cette variable est ignorée si `AdressesLocales` est différent de `LISTE`.
  - si c'est un nom de fichier, le fichier doit contenir les noms des machines, un par ligne.
  - les noms de machines ne doivent pas être qualifiés.

3.6.6 AdressesInternes

- type : mot-clef `RIEN`, `ETOILE_DOMAINE` ou `LISTE`
- signification : spécifie quelles sont les adresses reconnues comme étant dans notre juridiction, c'est-à-dire pour lesquelles cette machine agit comme « redistributeur » de courriers.
  Une connexion SMTP directe (sans utiliser de MX en particulier) est établie avec les machines « internes ».
  La signification des mot-clefs est donnée par le tableau :

Mot-clef	Signification
`RIEN`	aucun courrier n'est interne. Tout courrier non local est considéré comme externe.
`ETOILE_DOMAINE`	tous les courriers adressés à `*.domaine` doivent être distribués.
`LISTE`	seuls les courriers adressés aux machines d'une certaine liste doivent être redistribués.

- exemples :
  - `RIEN` pour le cas d'une feuille (avec ou sans boîtes aux lettres locales) : tout ce qui n'est pas local doit être envoyé à notre relais.
  - `RIEN` pour le cas d'un d'un serveur de laboratoire qui ne redistribue aucun courrier, mais qui les garde tous localement.
  - `ETOILE_DOMAINE` pour le serveur d'un laboratoire dans lequel tout courrier adressé à une station doit être transmis (et non stocké localement).
  - `LISTE` pour un *mailhost* d'un domaine, qui transmet le courrier pour certaines stations et garde en local tous les autres.
- remarques :
  - cette variable interagit fortement avec `AdressesLocales`. Toutes les combinaisons ne sont pas significatives.
  - si cette variable contient `LISTE`, la variable `ListeAdressesInternes` doit contenir la liste des stations pour lesquelles cette station conserve le courrier.

3.6.7 ListeAdressesInternes

- type : liste de noms de machines, ou nom de fichier (si le premier caractère est un « `/` »).
- signification : si la variable `AdressesInternes` = `LISTE`, cette variable contient la liste explicite des machines pour lesquelles le courrier doit être transmis par connexion SMTP directe.
- exemples :

  - `bambi picsou choupette`
  - `/usr/local/etc/mail/interne`
- remarques :
  - cette variable est ignorée si `AdressesInternes` est différent de `LISTE`.
  - si c'est un nom de fichier, le fichier doit contenir les noms des machines, un par ligne.
  - les noms de machines ne doivent pas être qualifiés.

3.6.8 MailhostEnInterne

- type : variable définie (contenu non vide, peu importe la valeur) ou non définie (contenu vide).
- signification : si cette variable est définie, la redistribution des courriers internes pour un domaine *x* se fait en ouvrant une connexion SMTP directe avec la machine `mailhost.`*x*. Si cette variable n'est pas définie, la redistribution vers *x* se fait en ouvrant une connexion SMTP directe avec *x*.
- exemples :
  - *mailhost* de l'UVSQ : variable définie, le *mailhost* envoie tout le courrier d'un laboratoire au *mailhost* correspondant.
  - *mailhost* de l'Institut Blaise Pascal : variable non définie, le *mailhost* envoie directement à la station concernée.
- remarque : cette variable n'est utile que si la variable `AdressesInternes` est différente de `RIEN`.

3.6.9 RelaisExterieur

- type : variable non définie (contenu vide) ou agent de transport + nom de relais vers l'extérieur.
- signification :
  - si cette variable n'est pas définie, cela signifie que cette machine route directement les courriers dans l'Internet (en se servant des MX) ou vers la passerelle BITNET adéquate.
  - si cette variable est définie, cela signifie que tout le courrier non local et non interne doit être envoyé (sans tenir compte des MX) à l'adresse spécifiée.
    Dans ce cas, cette variable contient le nom de l'agent de transport (*mailer*) à utiliser, ainsi que le nom du relais. Le plus souvent, l'agent de transport sera `smtp`. Avec la version 8 de `sendmail`, il faut entourer le relais entre crochets afin de ne pas utiliser les MX.
- exemples :
  - *mailhost* de site : variable non définie
  - *mailhost* de laboratoire dans le domaine `uvsq.fr` : `smtp.[mailhost.uvsq.fr]` (la machine centrale de courrier de l'UVSQ)
  - *feuille* du laboratoire `prism` dans le domaine `uvsq.fr` : `smtp.[mailhost.prism.uvsq.fr]` (tout courrier non local doit être envoyé au *mailhost*).
  - site raccordé ne disposant que d'une connectivité UUCP avec le reste du monde : `uucp.soleil` (le nœud `soleil` est le relais).
  - un effet secondaire de cette variable est de fixer le temps de conservation des courriers dans la file d'attente à 5 jours lorsque la machine est un relais de messagerie (au lieu de 3 jours par défaut).
- remarques :
  - l'absence de `RelaisExterieur` implique de savoir gérer les MX. Seule la version 8 de `sendmail` permet de gérer facilement les deux modes (avec ou sans MX) avec un seul exécutable.
    Sauf à utiliser la version 8 de `sendmail`, l'absence de `RelaisExterieur` est donc incompatible avec une redistribution interne (`AdressesInternes` différent de `RIEN`).
  - le relais peut être un alias dans le DNS, ce qui permet d'avoir un nom comme `mailhost.uvsq.fr`.
  - avec la version 8 de `sendmail`, il est possible d'indiquer plusieurs relais à la place d'un seul. Ainsi, si le premier relais est indisponible, `sendmail` tentera de dialoguer avec le deuxième, et ainsi de suite. Il suffit de séparer les relais par des caractères deux-points. Par exemple, le LIP6 (Laboratoire d'Informatique de

Paris VI) a deux[12] relais de messagerie : `isis.lip6.fr` et `osiris.lip6.fr`. Il y a donc deux RR de type MX. Les autres machines du LIP6 doivent donc avoir `RelaisExterieur` égal à :
`smtp.[isis.lip6.fr]:[osiris.lip6.fr]`
De la même manière, tout mailhost de laboratoire de Jussieu envoie normalement son courrier à la machine `mailhost.jussieu.fr`. Toutefois, si cette machine est indisponible, elle peut l'envoyer à `mailhost.uvsq.fr`, ce qui va de pair avec le traitement des MX (voir 1.6, page 16, ainsi que 3.6.10, page 88).
- Si l'agent de transport choisi est `uucp`, il ne faudra pas oublier de définir la variable `MailerUucp` (voir 3.6.24, page 95).

3.6.10 TableRoutages

- type : variable définie (méthode d'accès + nom de fichier) ou non définie (contenu vide).
- signification : si cette variable est définie, elle indique une méthode d'accès (`dbm` ou `hash -N`) ainsi qu'un nom de fichier contenant des directives de routages spécifiques à certains domaines, ou la méthode `ldap` et le filtre de recherche.
- exemples :
  - dans le cas général, cette variable n'est pas définie
  - `hash -N /usr/local/etc/mail/routages`
    Cela signifie que le fichier `/usr/local/etc/mail/routages` est un fichier compilé avec la méthode d'accès `hash -N`, donc en format *db*.
  - `dbm /usr/local/etc/mail/routages`
    Ce deuxième exemple montre la deuxième méthode d'accès, la méthode *dbm* qui ne doit être utilisée que si `sendmail` n'a pas pu être compilé avec la librairie `libdb`.
  - dans le cas d'une requête LDAP, il faut spécifier le filtre de recherche. Par exemple, pour utiliser le schéma par défaut de `sendmail` (voir 2.6, page 50) :

```
ldap -k "(& (objectClass=sendmailMTAMapObject)
            (sendmailMTAMapName=routages)
            (| (sendmailMTACluser=${sendmailMTACluster})
               (sendmailMTAHost=$j))
            (sendmailMTAKey=%0))"
           -1 -v sendmailMTAMapValue
```

    Attention toutefois : le kit attend que tout soit sur la même ligne.
    Il faut en outre veiller à ce que les options par défaut de LDAP soient définies à l'aide de la variable `ParametresLDAP`.
  - contenu du fichier (voir la syntaxe dans les remarques suivantes) :

frmug.fr.net:	uucp.frmug
machine-speciale.prism.uvsq.fr:	smtp.[machine-speciale.prism.uvsq.fr]
vieux-labo.uvsq.fr:	smtp.[mailhost.nouveau-labo.uvsq.fr]
site.bugge.edu:	buggysmtp.site.bugge.edu
site.non.enregistre.dans.dns:	smtp.[193.51.24.1]

    Dans cet exemple, la première ligne spécifie que tout courrier envoyé à `frmug.fr.net` (ou en réalité `*.frmug.fr.net`) est à envoyer, avec l'agent de transport `uucp` (défini avec la variable `MailerUucp`, voir 3.6.24, page 95), directement au relais `frmug`.
    La deuxième ligne décrit un cas particulier : alors que tout le courrier pour `prism.uvsq.fr` est envoyé au *mailhost* du laboratoire, le courrier destiné à `machine-particuliere` est directement envoyé à cette machine. Notez les crochets qui, en version 8 de `sendmail`, ont pour effet d'ignorer les MX.
    La troisième ligne donne l'exemple d'un laboratoire qui veut changer de nom. Tout courrier adressé à ce vieux nom est redirigé sur le nouveau laboratoire (ce qui peut être aussi assuré par un alias dans le DNS).

[12] Deux sur le campus, mais également deux autres relais extérieurs.

La quatrième ligne décrit un problème quelquefois rencontré avec la version 8 de `sendmail`. Puisque cette nouvelle version utilise ESMTP, il arrive que certains sites (qui violent ainsi la RFC 821) rompent immédiatement la connexion. Pour éviter cette rupture, tout courrier adressé à un tel site (traitement au cas par cas) est envoyé par l'agent de transport `buggysmtp` (pré-défini dans `sendmail.cf`) qui utilise SMTP et non ESMTP.

La dernière ligne donne l'exemple d'un site non encore enregistré dans le DNS. Tout courrier doit être en fait envoyé à l'adresse IP (adresse numérique) spécifiée.

- Ce fichier peut également servir à améliorer le traitement des MX (voir 1.6, page 16) en réception. Par exemple, la table de routages de `soleil.uvsq.fr` contient une ligne par sous-domaine de `jussieu.fr` :

```
ccr.jussieu.fr:    smtp.[mailhost.ccr.jussieu.fr]
lptl.jussieu.fr:   smtp.[mailhost.lptl.jussieu.fr]
...
```

  Ainsi, si la machine `shiva.jussieu.fr` (le MX « normal » de `jussieu.fr`) est indisponible, les courriers peuvent quand même être distribués à l'intérieur de Jussieu, le service n'est pas interrompu.

  Dans l'autre sens (des laboratoires vers l'extérieur), la variable `RelaisExtérieur` (voir 3.6.9, page 87) permet d'obtenir le même effet.

- remarques :
  - cette variable ne peut être définie que si la version 8 de `sendmail` est utilisée.
  - chaque ligne du fichier des routages doit avoir le format :

    domaine: *agent.relais*

    Toute adresse de la forme *domaine* ou *.*domaine* sera donc routée via l'agent de transport *agent* vers le *relais*. Les agents de transports définis dans le kit de Jussieu sont :

Agent	**Signification**
`local`	remise physique
`prog`	courrier envoyé à un programme
`smtp`	ESMTP
`buggysmtp`	SMTP simple (sans utiliser le mot-clef `EHLO` qui heurte les implémentations non conformes)
`uucp`	UUCP (si la variable `MailerUucp` est définie)

    Dans les agents `smtp` et `buggysmtp`, le fait d'entourer le relais entre crochets indique qu'il ne faut pas utiliser les MX.
  - ce fichier est une *map* de la version 8. Pour le compiler, faire (le chemin n'est qu'un exemple) :

```
/usr/lib/sendmail -bi -oA/usr/local/etc/mail/routages
```

    Cette astuce est basée sur l'option `-bi`, qui demande la reconstruction du fichier des aliases (comme le programme `newaliases`), et l'option `-oA` qui spécifie un nouveau chemin pour ce fichier. La compilation est effectuée en format *db* par défaut si `sendmail` est compilé avec la librairie `libdb`, ou en format *dbm* sinon.
  - ce mécanisme de table des routages court-circuite tout le mécanisme de sélection des adresses locales, internes ou externes. Il faut donc l'utiliser avec soin.

3.6.11 ListeNoire

- type : variable définie (méthode d'accès + nom de fichier) ou non définie (contenu vide).
- signification : si cette variable est définie, elle active les règles anti-*spam*, et spécifie le nom d'un fichier contenant la liste des *spammeurs* ou des exceptions.
- exemples :

  - dans le cas d'une organisation où seule une machine est autorisée à recevoir des courriers, les autres machines n'ont pas à définir cette variable ;
  - `hash -N /usr/local/etc/liste-noire`
    Cela signifie que le fichier `/usr/local/etc/liste-noire` est un fichier compilé avec la méthode d'accès `hash -N`, donc en format *db*.
  - contenu du fichier (voir la syntaxe en 3.5.4, page 72) :

    ```
    cyberpromo.com          SPAM
    gentil.cyberpromo.com   OK
    204.137.223             SPAM
    mechant@aol.com         SPAM
    domaine.fr              LOCAL
    eudora.baladeur.edu     LOCAL
    ```

    Dans cet exemple, la première ligne interdit tout courrier et toute connexion provenant du domaine `cyberpromo.com`.
    La deuxième ligne décrit un cas particulier : seule une machine particulière dans ce domaine peut nous envoyer du courrier.
    La troisième ligne interdit toute connexion en provenance d'une adresse IP commençant par 204.137.223.
    La quatrième ligne interdit les courriers en provenance d'un utilisateur particulier.
    La cinquième ligne autorise le relayage par le client spécifié.
  - dans le cas d'une requête LDAP, il faut spécifier le filtre de recherche. Par exemple, pour utiliser le schéma par défaut de `sendmail` (voir 2.6, page 50) :

    ```
    ldap -k "(& (objectClass=sendmailMTAMapObject)
                (sendmailMTAMapName=listenoire)
                (| (sendmailMTACluser=${sendmailMTACluster})
                   (sendmailMTAHost=$j))
                (sendmailMTAKey=%0))"
               -1 -v sendmailMTAMapValue
    ```

    Attention toutefois : le kit attend que tout soit sur la même ligne.
    Il faut en outre veiller à ce que les options par défaut de LDAP soient définies à l'aide de la variable `ParametresLDAP`.
- remarques :
  - voir la section 3.5.4 (page 71) consacrée à la lutte anti-*spam*.
  - cette variable ne peut être définie que si la version 8 de `sendmail` est utilisée.
  - ce fichier est une *map* de la version 8. Pour le compiler, faire (le chemin n'est qu'un exemple) :

    ```
    cd /usr/local/etc
    makemap -N hash liste-noire < liste-noire
    ```

  - il n'y a pas besoin d'arrêter et de redémarrer `sendmail` après modification de ce fichier.
  - il y a deux cas particuliers, correspondants aux deux lignes :

    ```
    NUMERIQUE       OK
    ```

    et

    ```
    RBL       SPAM domaine1 : domaine2 : ...
    ```

  - tout domaine routé explicitement, c'est à dire cité dans le fichier `TableRoutages` (voir 3.6.10, page 87) est considéré comme local, c'est à dire comme étant « relayable ».

3.6.12 CodeErreur

- type : variable définie (valeur numérique ou bi-numérique) ou non définie (contenu vide).
- signification : cette variable n'a de signification que si la variable `ListeNoire` est définie. Dans ce cas, c'est le (ou les) code(s) d'erreur retourné(s) pour tous les rejets dûs aux règles *anti-spam*.

- valeur par défaut : 451
- exemples :
  - `451` (ou non définie)
    Dans ce cas, tout rejet sera temporaire, le client devra essayer à nouveau. Cela présente deux avantages :
    - si un rejet est injustifié, par exemple parce que les serveurs DNS ne répondent pas assez vite, le client essayera à nouveau ;
    - si un rejet est justifié, le *spammeur* sera embêté car le courrier restera dans sa file d'attente

    En revanche, cela présente deux inconvénients :
    - si l'expéditeur est de toute bonne foi, il n'aura connaissance de son erreur que quelques jours plus tard, lorsque le courrier sera resté plus d'une certaine période dans la file d'attente du client SMTP.
    - les logs peuvent grossir si le client SMTP réïtère ses requêtes trop souvent.
  - `551`
    Tout rejet sera définitif. Dans ce cas, il faut surveiller attentivement les logs (voir 3.5.4, page 75) pour vérifier qu'il n'y a pas de rejet abusif.
  - `551 451`
    Les rejets dûs à des causes « fiables » (par exemple si une adresse est trouvée dans une liste noire) sont définitifs, les rejets dûs à des causes « non fiables » sont temporaires.
- remarques :
  - voir la section 3.5.4 (page 71) consacrée à la lutte anti-*spam*.
  - cette variable ne peut être définie que si la variable `ListeNoire` est définie.

3.6.13 URL

- type : chaîne de caractères vide, ou contenant une URL.
- signification : cette variable n'a de signification que si la variable `ListeNoire` est définie. Dans ce cas, elle contient une URL à renvoyer avec les messages de rejets dûs aux règles *anti-spam*.
- valeur par défaut : `http://www.prism.uvsq.fr/~pda/kit-jussieu/rejet`
- exemples :
  - (chaîne vide)
    Dans ce cas, aucune URL ne sera renvoyée avec le message de rejet.
  - `http://www.votre-site.fr/rejet`
    Tous les messages dûs aux règles anti-spam contiendront l'URL ci-dessus.
- remarques :
  - voir la section 3.5.4 (page 71) consacrée à la lutte anti-*spam*.
  - cette variable ne peut être définie que si la variable `ListeNoire` est définie.

3.6.14 ReecritureAdressesLocales

- type : variable non définie (contenu vide) ou définie (nom de domaine).
- signification : signature des courriers émis localement.
- exemples :
  - pour une feuille, qui envoie tout à son *mailhost*, cette variable n'est pas définie
  - pour `soleil.uvsq.fr`, cette variable contient `uvsq.fr`
  - pour le laboratoire PRiSM de l'UVSQ qui veut masquer le nom de machine, cette variable contient : `prism.uvsq.fr`
  - pour le *mailhost* d'un laboratoire qui ne veut pas masquer le nom des machines, cette variable contient : `machine.labo.uvsq.fr`
- remarques :
  - cette variable n'a de sens que si `AdressesLocales` est différent de `RIEN`
  - seules les adresses d'expéditeur classées comme *locales* (voir variable `AdressesLocales`) sont réécrites.

3.6.15 ReecritureAdressesInternes

- type : variable non définie (contenu vide) ou définie (nom de domaine).
- signification : signature des courriers internes, c'est-à-dire des courriers en provenance de machines dans notre juridiction.
- exemples :
  - pour une feuille, qui envoie tout à son *mailhost*, cette variable n'est pas définie
  - pour `soleil.uvsq.fr`, cette variable n'est pas définie (il n'y a pas de réécriture, sauf pour les courriers locaux)
  - pour *mailhost* du laboratoire PRiSM de l'UVSQ, où on désire masquer le nom de machine, cette variable contient : `prism.uvsq.fr`
  - pour le *mailhost* d'un laboratoire qui ne veut pas masquer le nom des machines, cette variable n'est pas définie.
- remarques :
  - cette variable n'a de sens que si `AdressesInternes` est différent de `RIEN`
  - seules les adresses d'expéditeur classées comme *internes* (voir variable `AdressesInternes`) sont réécrites.

3.6.16 RevAliases

- type : variable définie (méthode d'accès + nom de fichier) ou non définie (contenu vide).
- signification : si cette variable n'est pas définie, aucune traduction du nom d'utilisateur n'est effectuée. Si elle est définie, elle indique une méthode d'accès (`nis`, fichier local en format `dbm` ou `hash -N`) ainsi que la localisation (map NIS ou chemin du fichier local) de la table de correspondance *login* → *Prénom.Nom*, ou encore la méthode `ldap` et le filtre de recherche.
- exemples :
  - pour une feuille, cette variable n'est vraisemblablement pas définie ;
  - pour un *mailhost* qui veut réécrire les adresses, cette variable peut contenir l'une des trois valeurs suivantes :
    - `hash -N /usr/local/etc/mail/revaliases`
      et le fichier correspondant :
      ```
      pda:   Pierre.David
      jt:    Jacky.Thibault
      ```
    - `dbm /usr/local/etc/mail/revaliases`
    - `nis mail.revaliases`
    - dans le cas d'une requête LDAP, il faut spécifier le filtre de recherche. Par exemple, pour utiliser le schéma par défaut de `sendmail` (voir 2.6, page 50) :
      ```
      ldap -k "(& (objectClass=sendmailMTAAliasObject)
                  (sendmailMTAAliasGrouping=revaliases)
                  (| (sendmailMTACluser=${sendmailMTACluster})
                     (sendmailMTAHost=$j))
                  (sendmailMTAKey=%0))"
                 -v sendmailMTAAliasValue
      ```
      Attention toutefois : le kit attend que tout soit sur la même ligne.
      Il faut en outre veiller à ce que les options par défaut de LDAP soient définies à l'aide de la variable `ParametresLDAP`.
- remarques :
  - cette variable ne peut être définie que si la version 8 de `sendmail` est utilisée.
  - la réécriture n'affecte que les adresses d'expéditeur. Les adresses de destinataire ne doivent pas être réécrites.
  - chaque ligne de ce fichier doit avoir le format :

    login`:` *nouvelle-adresse*

    Il est conseillé, pour *nouvelle-adresse*, de choisir la convention *Prénom*`.`*Nom*, en respectant les majuscules et

en remplaçant chaque caractère non alphabétique par un tiret.
- ce fichier est une *map* de la version 8. Pour le compiler, faire (le chemin n'est qu'un exemple) :

```
/usr/lib/sendmail -bi -oA/usr/local/etc/mail/revaliases
```

La compilation se fera en format *db* si la librairie `libdb` a été compilée avec `sendmail` 8, ou en format *dbm* sinon.
- il est également possible de réécrire des adresses non locales. Par exemple :

```
pda@prism.uvsq.fr   Pierre.David@uvsq.fr
```

Dans ce cas, la station sur laquelle réside ce fichier remplacera l'adresse de l'expéditeur, c'est-à-dire `pda@prism.uvsq.fr`, pour donner `Pierre.David@uvsq.fr`.
L'utilisation est typiquement une base de réécriture centralisée au niveau d'une machine unique, ce qui permet d'avoir des adresses de la forme *Prénom*.*Nom*@*domaine*, même si le domaine est constitué d'un grand nombre de stations.
Attention : dans ce cas, l'astuce basée sur l'option `-bi` ne peut servir pour compiler ce fichier. Le format ne doit pas comporter de caractère « `:` » comme dans le cas précédent, mais les deux parties doivent être séparées par au moins un espace. La compilation de ce fichier se fait alors avec l'utilitaire `makemap` :

```
makemap -N hash revaliases < revaliases
```

Comme pour le fichier des routages, le fichier résultat est suffixé par « `.db` ». On peut utiliser `dbm` à la place de `-N hash` si la variable a été déclarée en format *dbm*.

3.6.17 Aliases

- type : nom de fichier ou méthode`:`fichier ou `ldap:`.
- signification : localisation du fichier des aliases.
- exemples :
  - sur Sun : `/etc/aliases`
  - sur HP-UX jusqu'en 9.0 : `/usr/lib/aliases`
  - avec la version 8 de `sendmail`, pour accéder aux aliases par les NIS : `nis:mail.aliases`
  - avec la version 8 de `sendmail`, pour accéder aux aliases par une base LDAP : `ldap:`.
    Il faut en outre veiller à ce que les options par défaut de LDAP soient définies à l'aide de la variable `ParametresLDAP`.
- remarques :
  - bien qu'il puisse être placé n'importe où, mieux vaut placer le fichier des aliases à l'endroit prévu par le constructeur.
  - ce fichier n'est pas nécessaire (il doit même être enlevé) si la station à configurer est une feuille sans répertoire de boîtes aux lettres local, c'est-à-dire lorsque `AdressesLocales` égale `RIEN`). La version 8 donnera un message d'erreur si le fichier existe et contient des aliases.
  - l'accès aux NIS n'est en général pas nécessaire lorsque la station à configurer est le serveur NIS maître (le fichier de référence est alors local).
  - si on souhaite utiliser un autre schéma que le schéma par défaut de `sendmail` (voir 2.6, page 50), il faut spécifier le filtre de recherche explicitement, comme par exemple avec (déclaration équivalent à la déclaration implicite) :

```
ldap -k "(& (objectClass=sendmailMTAAliasObject)
            (sendmailMTAAliasGrouping=aliases)
            (| (sendmailMTACluser=${sendmailMTACluster})
               (sendmailMTAHost=$j))
            (sendmailMTAKey=%0))"
           -v sendmailMTAAliasValue
```

Attention toutefois : le kit attend que tout soit sur la même ligne.

3.6.18 SendmailSt

- type : nom de fichier
- signification : localisation du fichier de comptabilité de `sendmail`.
- exemples :
  - sur Sun : `/etc/sendmail.st`
  - sur HP : `/usr/lib/sendmail.st`
  - sur FreeBSD : `/etc/mail/statistics`
- remarques :
  - bien qu'il puisse être placé n'importe où, mieux vaut placer le fichier de comptabilité à l'endroit prévu par le constructeur.
  - l'utilitaire `mailstats` permet d'obtenir les statistiques en clair. Vous pouvez utiliser la version du constructeur ou la version qui vient avec les sources de la version 8 pour une meilleure lisibilité.

3.6.19 SendmailHf

- type : nom de fichier
- signification : localisation du fichier d'aide de `sendmail`.
- exemples :
  - sur Sun : `/usr/lib/sendmail.hf`
  - sur HP-UX 10 : `/usr/share/lib/sendmail.hf`
  - sur FreeBSD : `/etc/mail/helpfile`
- remarques :
  - bien qu'il puisse être placé n'importe où, mieux vaut placer le fichier d'aide à l'endroit prévu par le constructeur.
  - ce fichier est utilisé dans deux occasions : sur requête `HELP` dans une session SMTP, ou dans le mode de test des règles de réécriture de `sendmail` (option `-bt`).

3.6.20 ParametresLDAP

- type : variable définie (chaîne d'options compatibles avec `ldapsearch`) ou non.
- signification : paramètres par défaut des requêtes LDAP.
- exemples :
  - pour un site qui n'utilise pas LDAP, cette variable n'est pas définie.
  - `-h ldap.domaine.fr -p 389 -b dc=domaine,dc=fr`
    pour préciser que le serveur est à l'adresse `ldap.domaine.fr`, que le port TCP utilisé est le port 389 et que le DN de base est `dc=domaine,dc=fr`.
- remarques :
  - cette variable n'a pas de valeur par défaut, il faut penser à l'activer.

3.6.21 Mqueue

- type : nom de répertoire
- signification : localisation du répertoire servant de file d'attente.
- exemples :
  - sur Sun : `/var/spool/mqueue`

  - sur HP-UX 9.x : `/usr/spool/mqueue`
- remarques :
  - bien qu'il puisse être placé n'importe où, mieux vaut placer ce répertoire à l'endroit prévu par le constructeur.
  - l'utilitaire `mailq` (qui est en fait un lien sur `sendmail`) affiche le contenu de cette file d'attente. Étant donné que les informations qui s'y trouvent peuvent être sensibles, il est possible de restreindre l'exécution de cette commande par le seul administrateur du système. Si vous utilisez la version 8, il vous suffit de localiser la ligne commençant par `Op` dans le fichier généré par le configurateur et d'y ajouter `restrictmailq`.
  - le temps de conservation des courriers dans la file d'attente est normalement de 3 jours (5 jours dans le cas d'un relais de messagerie, voir 3.6.9, page 86). Plus le délai est long, plus les erreurs, s'il y en a, sont signalées tardivement à l'utilisateur. En revanche, un délai long permet d'être plus tolérant envers des sites ayant des dysfonctionnements pendant des périodes prolongées, comme par exemple dans le cas de week-ends ou de ponts... Il est à noter que :
    - la variable `FichierInclude` (voir 3.6.27, page 96) permet de spécifier une autre valeur pour le temps de conservation : voir en particulier l'exemple en 3.5.7 (page 81),
    - il est possible d'indiquer à `sendmail` qu'il doit envoyer un message d'avertissement lorsqu'un courrier est resté bloqué dans la file d'attente pendant plus d'un certain temps (par exemple une heure) avec l'option de `sendmail` :
      ```
      O Timeout.queuewarn=1h
      ```
      Cette option peut également être ajoutée dans un fichier inclus grâce à la variable `FichierInclude`.

3.6.22 TailleMaximum

- type : variable non définie, ou vide, ou ayant une valeur numérique
- signification : taille maximum, en octets, des messages acceptés par `sendmail`, aussi bien en local que via SMTP.
- exemples :
  - pour ne pas limiter la taille des messages, il faut initialiser cette variable à une chaîne vide ;
  - pour limiter la taille des messages à 2 Mo, il faut mettre 2097152 ;
  - pour accepter la limite par défaut (16 Mo), il ne faut pas modifier cette variable.
- remarques :
  - la limite par défaut est fixée à 16 Mo.

3.6.23 MailerLocal

- type : liste
- signification : paramètres du *mailer* local.
- exemples :
  - sur HP-UX : `/bin/rmail DFMPlms rmail -d $u`
  - sur AIX 3.2 : `/bin/bellmail DFMPlmns mail $u`
  - sur beaucoup de systèmes : `/bin/mail DFMPlmnrs mail -d $u`
- remarques :
  - la meilleure manière de déterminer ces paramètres est encore de consulter le fichier `sendmail.cf` qui est livré avec votre système (consulter la ligne commençant par `Mlocal`).
  - du fait que cette variable contient un caractère `$`, vous devez utiliser des quotes simples (apostrophes) au lieu des quotes doubles (guillemets) pour délimiter la valeur de cette variable. Ceci est dû à l'interprétation des `$` par le Shell.

3.6.24 MailerUucp

- type : variable définie (liste) ou non
- signification : paramètres du *mailer* UUCP.
- exemples :
  - pour un site qui n'utilise pas UUCP, cette variable n'est pas définie.
  - `'/usr/bin/uux DFMhmsu uux - -r $h!rmail ($u)'`
- remarques :
  - la définition de cette variable active les règles de réécriture propres au *mailer* UUCP.
  - cette variable n'est utile que si l'une au moins des deux variables `TableRoutages` (voir 3.6.10, page 87) ou `RelaisExterieur` (voir 3.6.9, page 86) est définie, car il n'y a en effet aucun autre moyen de router les messages vers une liaison UUCP.

3.6.25 SansDNS

- type : variable définie (contenu non vide, peu importe la valeur) ou non définie (contenu vide).
- signification : la version 8 de `sendmail`, quand le serveur de noms n'est pas actif, considère qu'il s'agit d'un dysfonctionnement temporaire et place le courrier en file d'attente. Ce comportement est génant si un site n'utilise pas le serveur de noms.
- exemples :
  - pour toute station correctement configurée et connectée, cette variable n'est pas définie.
  - dans le passé, les machines du domaine `ensinfo.uvsq.fr` (réseau d'enseignement des deuxième et troisième cycles d'informatique à l'UVSQ) ne pouvaient pas utiliser tous les services Internet, et en particulier pas le DNS. Cette variable était donc initialisée à une valeur quelconque.
- remarques :
  - cette configuration, bien que prévue, n'est pas conseillée.
  - si on n'utilise pas le serveur de noms, mais seulement les NIS ou le fichier `/etc/hosts`, il faut mettre en premier les noms qualifiés. Par exemple :

```
193.51.24.1   soleil.uvsq.fr              soleil     mailhost.uvsq.fr
193.51.24.1   romuald.ensinfo.uvsq.fr     romuald    mailhost.ensinfo.uvsq.fr
193.51.26.2   athanase.ensinfo.uvsq.fr    athanase
etc.
```

3.6.26 SansCanonisation

- type : variable définie (contenu vide) ou non définie (contenu non vide, peu importe la valeur).
- signification : dans certaines conditions très particulières, lorsqu'on sait que les adresses fournies en entrée sont correctes, il n'est pas nécessaire que `sendmail` procède à la canonisation des adresses, ce qui ralentit le processus de distribution et alourdit la charge de la machine.
- exemples :
  - dans quasiment tous les cas « classiques », cette variable n'est pas définie.
  - cette variable est définie pour certains sites qui font de la distribution en deux temps, comme par exmple un site spécialisé dans la gestion de listes de diffusions. Dans ce cas, le processus `sendmail` qui reçoit les messages canonise les adresses, communique les messages au gérant de liste, qui appelle de nouveau `sendmail` de multiples fois ; pour ce deuxième temps, il n'y a pas besoin de canoniser les adresses.
- remarques :
  - réfléchissez-bien avant d'utiliser cette option ! Il y a de nombreuses implications pour les autres variables,

comme par exemple `ReecritureAdressesLocales` ou d'autres.

3.6.27 FichierInclude

- type : variable définie (nom de fichier) ou non définie (contenu vide).
- signification : si cette variable est définie, elle contient le nom d'un fichier à inclure dans le `sendmail.cf` généré. Cela peut être intéressant pour ajouter des options, des macros ou des règles spécifiques à un site.
- exemples :
  - pour la majorité des sites, cette variable n'a pas besoin d'être définie ;
  - `toto.cf`
    Cela signifie que le fichier `toto.cf` contient des lignes à ajouter au fichier `sendmail.cf` généré.
  - ce fichier peut être utilisé pour réaliser un filtrage anti-spam lorsque le relais de messagerie n'en fait pas. Voir 3.7.4, page 116.
- remarques :
  - le fichier doit exister.
  - les définitions (mailers, macros, options, etc.) contenues dans ce fichier priment sur les définitions antérieures, car le fichier est inclus après toutes les définitions, et juste avant les règles de réécriture.
  - les commentaires, le cas échéant, sont introduits par les deux caractères « `_#` ».
  - un exemple figure en 3.5.7 (page 81).

3.7 Exemples de configurations

Cette section donne quelques exemples de configuration (réels ou imaginaires). Ce sont :

- Bureau des Longitudes
  Cet exemple illustre une architecture de courrier très simple, très homogène et petite.
- Jussieu et Université de Versailles – Saint Quentin en Yvelines
  Cet exemple est un exemple de configuration complexe à 4 champs.
- Institut Blaise Pascal
  Cet exemple montre une configuration très hiérarchisée.
- Maison
  Cet exemple montre la configuration d'un site « individuel », c'est-à-dire d'un pécé, par exemple, chez un particulier connecté par modem à un fournisseur de connectivité IP.
  5.4
- Utilisation de LDAP
  Cet exemple fictif montre un fichier de configuration pour une machine allant chercher des informations sur un serveur LDAP.

Pour chaque exemple, l'architecture est brièvement décrite, puis quelques exemples de fichiers de configuration de stations sont listés, enfin le fichier de règles du site est donné.

3.7.1 Bureau des Longitudes

Architecture

Le Bureau des Longitudes est un exemple simple. Une machine (`gentiane.bdl.fr`) reçoit tout le courrier et le stocke dans le répertoire des boîtes aux lettres, exporté en NFS vers les autres stations du réseau. Toutes les autres machines sont des feuilles, elles ne reçoivent aucun courrier local, elles reroutent tout vers `gentiane`. Dans le DNS, les MX de toutes les stations ainsi que du domaine `bdl.fr` pointent sur `gentiane.bdl.fr`.

Fichiers de configuration

Dans cette architecture, l'administrateur d'une station doit juste spécifier si sa station est le *mailhost* ou non.

Fichier `gentiane.config` (mailhost du domaine) :

```
Host='gentiane'
Mailhost='o'
RevAliases='hash -N /usr/local/etc/revaliases'

Aliases='/usr/lib/aliases'
SendmailSt='/usr/lib/sendmail.st'
SendmailHf='/usr/lib/sendmail.hf'
MailerLocal='/bin/mail DFMPlmnrs mail -d $u'
Mqueue='/usr/spool/mqueue'
```

Fichier `centauree.config` (exemple de feuille) :

```
Host='centauree'
Mailhost='n'

Aliases='/usr/lib/aliases'
SendmailSt='/usr/lib/sendmail.st'
SendmailHf='/usr/lib/sendmail.hf'
MailerLocal='/bin/mail DFMPlmnrs mail -d $u'
Mqueue='/usr/spool/mqueue'
```

Script de traduction

```
#
# Regles correspondant au site bdl.fr
#
# Un mailhost de domaine
# Quelques stations dans le domaine partageant un spool commun
# Signature homogene : Prenom.Nom@bdl.fr
#

#
# Le nom du site. Utile pour les autres organismes qui veulent
# s'inspirer de ce fichier.
```

```
#

# SITE=votre-domaine.fr
SITE=bdl.fr

#
# Variables disponibles pour les fichiers de config (ceux
# fournis aux utilisateurs)
#
#       Host=
#       Mailhost= o/n
#       RevAliases= (fichier)
#       + les variables classiques "systemes" (Aliases, etc.)
#

case "$Mailhost" in
    o)
        Domaine=$SITE
        ListeDomaines="$Domaine"
        AdressesLocales=TOUT_DOMAINE
        AdressesInternes=RIEN
        RelaisExterieur=
        MailhostEnInterne=
        TableRoutages=
        ReecritureAdressesLocales=$Domaine
        ReecritureAdressesInternes=
        ;;
    n)
        Domaine=$SITE
        ListeDomaines="$Domaine"
        AdressesLocales=RIEN
        AdressesInternes=RIEN
        RelaisExterieur="smtp.[mailhost.$Domaine]"
        MailhostEnInterne=
        TableRoutages=
        ReecritureAdressesLocales=
        ReecritureAdressesInternes=
        ;;
    *)
        echo "Variable Mailhost mal definie" >&2
        exit 1
        ;;
esac
```

3.7.2 Jussieu et UVSQ

Architecture

Les cas de Jussieu (domaine `jussieu.fr`) et de l'Université de Versailles – Saint Quentin en Yvelines (domaine `uvsq.fr`) sont très similaires. Chaque machine, hormis une ou deux machines de service, portent des noms à quatre champs (exemple : `cezanne.prism.uvsq.fr`). Tout le courrier est dirigé (par des MX) sur une machine centrale de courrier (`shiva.jussieu.fr` ou `soleil.uvsq.fr`), qui redirige ensuite vers les *mailhosts* des laboratoires, interface unique avec le *mailhost* du campus. Chaque laboratoire a sa propre gestion des courriers, mais envoie tous les courriers qui ne sont pas dans le laboratoire au *mailhost* du campus.

Fichiers de configuration

Dans cette architecture, il y a un peu plus de paramètres pour chaque laboratoire :

Si la station à configurer est un *mailhost* :

- le nom de la station
- son laboratoire (c'est-à-dire le troisième champ du nom complet)
- est-ce que le courrier est gardé sur le *mailhost* ou redistribué aux autres machines ?
- si le courrier doit être gardé sur le *mailhost*, doit-il l'être pour l'ensemble du laboratoire ou pour quelques stations seulement ?
- le *mailhost* doit-il signer en retirant le nom de machine ou non ?
- le *mailhost* doit-il réécrire les adresses *login* en *Nom.Prénom* ou ne pas les réécrire ?
- localisation du fichier `aliases`
- localisation du fichier `statistics`
- version 8 de `sendmail` ou non.

Si la station à configurer est une *feuille* :

- le nom de la station
- son laboratoire (c'est-à-dire le troisième champ du nom complet)
- la station renvoit-elle tout au *mailhost* ou conserve-t-elle certains courriers localement ?
- localisation du fichier `aliases`
- localisation du fichier `statistics`
- version 8 de `sendmail` ou non.

Les fichiers de configuration sont directement issus de l'ancienne version du configurateur. Si un administrateur de laboratoire a configuré sa station avec l'ancien configurateur, il peut reprendre son fichier de variables inchangé et le passer dans le nouveau configurateur, muni des règles de Jussieu.

Par exemple, `bambi` est le *mailhost* du laboratoire LPTL à Jussieu :

```
#
# Configuration du sendmail des laboratoires de jussieu.fr
#
# Auteurs :
#   Jacky Thibault (jt@ccr.jussieu.fr)
#   Pierre David (Pierre.David@prism.uvsq.fr)
#
# Historique
#   91/04/03 : pda/jt : conception
#   91/11/28 : jt/pda : disparition des cas speciaux CCR et LPTHE
#   92/07/17 : jt/pda : simplification
#   92/11/26 : pda    : adaptation pour sendmail V8 et revaliases
#   96/02/08 : jt/pda : adaptation pour sendmail 8.7 et systemes differents
#

#
# Configuration de bambi.lptl.jussieu.fr :
# - c'est une machine Solaris 2.4
# - c'est un mailhost qui garde tous les courriers a destination du labo
# - pas de revaliases
#
```

```
#
# Changer les valeurs des variables ci-dessous en fonction de votre site.
# Attention : ne pas laisser d'espaces de part et d'autre du signe "="
# dans les affectations.
#

###############################################################################
# Parametres du site
###############################################################################

#
# Host
#
# Mettez ici le nom de la machine sans le nom de domaine
# Exemple : "bambi" pour "bambi.lptl.jussieu.fr".
#
Host='bambi'

#
# Labo
#
# Mettez ici le nom de votre laboratoire.
# Ca doit correspondre a la partie "laboratoire" du nom du domaine.
# Exemple : "lptl" pour "lptl.jussieu.fr"
#
Labo='lptl'

#
# Mailhost
#
# Indiquez ici si ce sendmail est destine a la machine servant de
# mailhost pour le laboratoire.
#
# Valeurs possibles :
#       o
#       n
#
Mailhost='o'

###############################################################################
# Variables pour le MAILHOST
# NE LES MODIFIER QUE POUR LA CONSTITUTION D'UN sendmail.cf D'UN mailhost
###############################################################################

#
# SpoolCommun
#
# Si "o", tout le courrier a destination du laboratoire (ou du groupe de
# machines defini par SpoolMachines) doit etre garde sur ce mailhost.
# Si "n", le courrier doit etre individuellement envoye a la machine du
# laboratoire.
#
# Valeurs possibles :
#       o
#       n
#
SpoolCommun='o'
```

```
#
# SpoolMachines
#
# Ensemble des machines pour lesquelles le courrier est garde sur le
# mailhost (soit parce que le mailhost est la machine ou tout le monde
# lit son courrier, soit parce que les autres machines montent le spool/mail
# par NFS).
# A initialiser seulement si SpoolCommun="o"
#
# Valeurs possible :
#       chaine vide si l'ensemble = tout le laboratoire
#       liste de machines separees par des espaces (incluant le mailhost)
#
# Exemple : SpoolMachines="bambi mickey jumbo picsou cendrillon"
#
SpoolMachines=''

#
# FromLabo
#
# Si "o", le nom du host est retire du champ "From: " pour ne laisser que
# le nom du laboratoire (ex : lptl.jussieu.fr au lieu de bambi.lptl.jussieu.fr)
# pour toutes les machines definies dans "FromLaboMachines"
# Si "n", le nom du host est laisse dans le champ "From: " pour tout le monde.
#
# Valeurs possibles :
#       o
#       n
#
FromLabo='o'

#
# FromLaboMachines
#
# Ensemble des machines pour lesquelles le mailhost doit retirer le host
# du champ "From: ".
# A initialiser seulement si FROMLABO="o"
#
# Valeurs possibles :
#       chaine vide si l'ensemble = tout le laboratoire
#       liste de machines separees par des espaces (incluant le mailhost)
#
# Exemple : FromLaboMachines="bambi mickey jumbo picsou cendrillon"
#
FromLaboMachines=''

###############################################################################
# Variables pour les feuilles
# NE LES MODIFIER QUE POUR LA CONSTITUTION D'UN sendmail.cf D'UNE feuille
###############################################################################

#
# ToutEnvoyer
#
# Si "o", le courrier local (ainsi que tous les autres courriers) doit etre
# envoye sur le mailhost et pas entrepose sur la feuille.
# Si "n", le courrier local doit rester sur cette machine.
#
# Attention ! La valeur mise ici doit etre homogene avec la politique
```

```
# adoptee dans votre laboratoire, definie sur le mailhost :
# si SpoolCommun="o" et (SpoolMachines="" ou la feuille est dans SpoolMachines)
#   alors ToutEnvoyer="o"
#   sinon ToutEnvoyer="n"
# fin si
#
# Valeurs possibles :
#       o
#       n
#
ToutEnvoyer='o'

###############################################################################
# Parametres du systeme
###############################################################################

#
# Pour determiner tous ces parametres, il faut localiser votre fichier
# sendmail.cf existant :
#       /etc/sendmail.cf                (SunOS, AIX, toutes machines avec V8)
#       /usr/lib/sendmail.cf            (autres...)
#

#
# Aliases
#
# Pour determiner la valeur :
#       Utilisez la commande "grep '^OA' sendmail.cf"
#
# Valeurs possibles :
#       /etc/aliases                    (SunOS, AIX, BSD 4.4, ...)
#       /etc/mail/aliases               (Solaris, HP-UX 10)
#       /usr/lib/aliases                (autres...)
#
Aliases='/etc/mail/aliases'

#
# SendmailSt
#
# Pour determiner la valeur :
#       Utilisez la commande "grep '^OS' sendmail.cf"
#
# Valeurs possibles :
#       /etc/sendmail.st                (SunOS, AIX)
#       /var/log/sendmail.st            (BSD 4.4)
#       /etc/mail/sendmail.st           (HP-UX 10, Solaris)
#       /usr/lib/sendmail.st            (autres...)
#
SendmailSt='/etc/mail/sendmail.st'

#
# SendmailHf
#
# Pour determiner la valeur :
#       Utilisez la commande "grep '^OH' sendmail.cf"
#
# Valeurs possibles :
#       /usr/share/lib/sendmail.hf      (HP-UX 10)
#       /usr/share/misc/sendmail.hf     (BSD 4.4)
```

```
#       /etc/mail/sendmail.hf           (Solaris)
#       /etc/sendmail.hf                (AIX)
#       /usr/lib/sendmail.hf            (autres...)
#
SendmailHf='/etc/mail/sendmail.hf'

#
# Mqueue
#
# Pour determiner la valeur :
#       Utilisez la commande "grep '^OQ' sendmail.cf"
#
# Valeurs possibles :
#       /var/spool/mqueue               (HP-UX 10, BSD 4.4, Solaris, AIX...)
#       /usr/spool/mqueue               (autres...)
#
Mqueue='/var/spool/mqueue'

#
# MailerLocal
#
# Pour determiner la valeur :
#       Utilisez la commande "grep '^Mlocal' sendmail.cf"
#
# Valeurs possibles :
#       '' si c'est une feuille qui envoie tout
#       'executable flags arguments'
#
# Exemples :
#       '/bin/rmail DFMPlms rmail -d $u'                (HP-UX)
#       '/bin/mail DFMPSflmnrs mail -d $u'              (Solaris 2.4)
#       '/usr/lib/mail.local DFMPSflmns mail.local -d $u' (Solaris 2.5)
#       '/bin/bellmail DFMPlmns mail $u'                (AIX <= 3.2)
#       '/usr/libexec/mail.local lsDFMrmn mail -d $u'   (BSD 4.4)
#       '/usr/bin/deliver DFMPlmnrs deliver $u'         (Linux)
#       '/bin/mail DFMPlmnrs mail -d $u'                (autres)
#
MailerLocal='/bin/mail DFMPSflmnrs mail -d $u'

#
# RevAliases
#
# Fichier contenant la base d'utilisateurs pour les reecritures
#       <nom-de-login>: <nom-en-clair>
# afin de reecrire les adresses sous la fome : Prenom.Nom@labo
# Ca n'est souvent utile que sur le mailhost.
#
# Valeurs possibles :
#       chaine vide : pas de traduction (implicite si pas V8)
#       methode + localisation du fichier contenant les aliases inverses
#
RevAliases=''
```

Et `peterpan` est une feuille dans ce laboratoire :

```
#
# Configuration du sendmail des laboratoires de jussieu.fr
#
```

```
# Auteurs :
#   Jacky Thibault (jt@ccr.jussieu.fr)
#   Pierre David (Pierre.David@prism.uvsq.fr)
#
# Historique
#   91/04/03 : pda/jt : conception
#   91/11/28 : jt/pda : disparition des cas speciaux CCR et LPTHE
#   92/07/17 : jt/pda : simplification
#   92/11/26 : pda    : adaptation pour sendmail V8 et revaliases
#   96/02/08 : jt/pda : adaptation pour sendmail 8.7 et systemes differents
#

#
# Configuration de peterpan.lptl.jussieu.fr :
# - c'est une machine HP-UX 9
# - c'est une feuille qui envoie tout
#

#
# Changer les valeurs des variables ci-dessous en fonction de votre site.
# Attention : ne pas laisser d'espaces de part et d'autre du signe "="
# dans les affectations.
#

###############################################################################
# Parametres du site
###############################################################################

#
# Host
#
# Mettez ici le nom de la machine sans le nom de domaine
# Exemple : "bambi" pour "bambi.lptl.jussieu.fr".
#
Host='peterpan'

#
# Labo
#
# Mettez ici le nom de votre laboratoire.
# Ca doit correspondre a la partie "laboratoire" du nom du domaine.
# Exemple : "lptl" pour "lptl.jussieu.fr"
#
Labo='lptl'

#
# Mailhost
#
# Indiquez ici si ce sendmail est destine a la machine servant de
# mailhost pour le laboratoire.
#
# Valeurs possibles :
#       o
#       n
#
Mailhost='n'

###############################################################################
# Variables pour le MAILHOST
```

```
# NE LES MODIFIER QUE POUR LA CONSTITUTION D'UN sendmail.cf D'UN mailhost
##############################################################################

#
# SpoolCommun
#
# Si "o", tout le courrier a destination du laboratoire (ou du groupe de
# machines defini par SpoolMachines) doit etre garde sur ce mailhost.
# Si "n", le courrier doit etre individuellement envoye a la machine du
# laboratoire.
#
# Valeurs possibles :
#       o
#       n
#
SpoolCommun='o'

#
# SpoolMachines
#
# Ensemble des machines pour lesquelles le courrier est garde sur le
# mailhost (soit parce que le mailhost est la machine ou tout le monde
# lit son courrier, soit parce que les autres machines montent le spool/mail
# par NFS).
# A initialiser seulement si SpoolCommun="o"
#
# Valeurs possible :
#       chaine vide si l'ensemble = tout le laboratoire
#       liste de machines separees par des espaces (incluant le mailhost)
#
# Exemple : SpoolMachines="bambi mickey jumbo picsou cendrillon"
#
SpoolMachines=''

#
# FromLabo
#
# Si "o", le nom du host est retire du champ "From: " pour ne laisser que
# le nom du laboratoire (ex : lptl.jussieu.fr au lieu de bambi.lptl.jussieu.fr)
# pour toutes les machines definies dans "FromLaboMachines"
# Si "n", le nom du host est laisse dans le champ "From: " pour tout le monde.
#
# Valeurs possibles :
#       o
#       n
#
FromLabo='o'

#
# FromLaboMachines
#
# Ensemble des machines pour lesquelles le mailhost doit retirer le host
# du champ "From: ".
# A initialiser seulement si FROMLABO="o"
#
# Valeurs possibles :
#       chaine vide si l'ensemble = tout le laboratoire
#       liste de machines separees par des espaces (incluant le mailhost)
#
```

```
# Exemple : FromLaboMachines="bambi mickey jumbo picsou cendrillon"
#
FromLaboMachines=''

###############################################################################
# Variables pour les feuilles
# NE LES MODIFIER QUE POUR LA CONSTITUTION D'UN sendmail.cf D'UNE feuille
###############################################################################

#
# ToutEnvoyer
#
# Si "o", le courrier local (ainsi que tous les autres courriers) doit etre
# envoye sur le mailhost et pas entrepose sur la feuille.
# Si "n", le courrier local doit rester sur cette machine.
#
# Attention ! La valeur mise ici doit etre homogene avec la politique
# adoptee dans votre laboratoire, definie sur le mailhost :
# si SpoolCommun="o" et (SpoolMachines="" ou la feuille est dans SpoolMachines)
#   alors ToutEnvoyer="o"
#   sinon ToutEnvoyer="n"
# fin si
#
# Valeurs possibles :
#       o
#       n
#
ToutEnvoyer='o'

###############################################################################
# Parametres du systeme
###############################################################################

#
# Pour determiner tous ces parametres, il faut localiser votre fichier
# sendmail.cf existant :
#       /etc/sendmail.cf                (SunOS, AIX, toutes machines avec V8)
#       /usr/lib/sendmail.cf            (autres...)
#

#
# Aliases
#
# Pour determiner la valeur :
#       Utilisez la commande "grep '^OA' sendmail.cf"
#
# Valeurs possibles :
#       /etc/aliases                    (SunOS, AIX, BSD 4.4, ...)
#       /etc/mail/aliases               (Solaris, HP-UX 10)
#       /usr/lib/aliases                (autres...)
#
Aliases='/usr/lib/aliases'

#
# SendmailSt
#
# Pour determiner la valeur :
#       Utilisez la commande "grep '^OS' sendmail.cf"
#
```

```
# Valeurs possibles :
#       /etc/sendmail.st                (SunOS, AIX)
#       /var/log/sendmail.st            (BSD 4.4)
#       /etc/mail/sendmail.st           (HP-UX 10, Solaris)
#       /usr/lib/sendmail.st            (autres...)
#
SendmailSt='/usr/lib/sendmail.st'

#
# SendmailHf
#
# Pour determiner la valeur :
#       Utilisez la commande "grep '^OH' sendmail.cf"
#
# Valeurs possibles :
#       /usr/share/lib/sendmail.hf      (HP-UX 10)
#       /usr/share/misc/sendmail.hf     (BSD 4.4)
#       /etc/mail/sendmail.hf           (Solaris)
#       /etc/sendmail.hf                (AIX)
#       /usr/lib/sendmail.hf            (autres...)
#
SendmailHf='/usr/lib/sendmail.hf'

#
# Mqueue
#
# Pour determiner la valeur :
#       Utilisez la commande "grep '^OQ' sendmail.cf"
#
# Valeurs possibles :
#       /var/spool/mqueue               (HP-UX 10, BSD 4.4, Solaris, AIX...)
#       /usr/spool/mqueue               (autres...)
#
Mqueue='/usr/spool/mqueue'

#
# MailerLocal
#
# Pour determiner la valeur :
#       Utilisez la commande "grep '^Mlocal' sendmail.cf"
#
# Valeurs possibles :
#       '' si c'est une feuille qui envoie tout
#       'executable flags arguments'
#
# Exemples :
#       '/bin/rmail DFMPlms rmail -d $u'                (HP-UX)
#       '/bin/mail DFMPSflmnrs mail -d $u'              (Solaris 2.4)
#       '/usr/lib/mail.local DFMPSflmns mail.local -d $u' (Solaris 2.5)
#       '/bin/bellmail DFMPlmns mail $u'                (AIX <= 3.2)
#       '/usr/libexec/mail.local lsDFMrmn mail -d $u'   (BSD 4.4)
#       '/usr/bin/deliver DFMPlmnrs deliver $u'         (Linux)
#       '/bin/mail DFMPlmnrs mail -d $u'                (autres)
#
MailerLocal=''

#
# RevAliases
#
```

```
# Attention : necessite V8 compile avec db
# Fichier contenant la base d'utilisateurs pour les reecritures
#       <nom-de-login>: <nom-en-clair>
# afin de reecrire les adresses sous la fome : Prenom.Nom@labo
# Ca n'est souvent utile que sur le mailhost.
#
# Valeurs possibles :
#       chaine vide : pas de traduction (implicite si pas V8)
#       methode + localisation du fichier contenant les aliases inverses
#
RevAliases=''
```

Script de traduction

```
#
# Regles correspondant au site jussieu.fr
#
# Un mailhost de domaine        mailhost.jussieu.fr
# Un mailhost par laboratoire   mailhost.labo.jussieu.fr
#       qui garde ou pas le courrier de tout (ou partiellement) le labo
# Des feuilles avec ou sans courrier local
#
# Signature homogene pour certains labos : Prenom.Nom@labo.jussieu.fr
# faite au niveau des mailhosts de labo.
#
# Les courriers depuis l'exterieur passent par shiva, puis sont retransmis
# vers le mailhost de laboratoire qui les reroute eventuellement
# via /etc/aliases ou selon le nom de machines
# Tout courrier provenant d'une machine d'equipe (feuille ou mailhost)
# est envoye au mailhost de laboratoire pour reecriture eventuelle du "From:".
#

#
# Le nom du site.
#

# SITE=votre-domaine.fr
# SITE=uvsq.fr
SITE=jussieu.fr

# le site de secours (vraisemblablement inutile hors de Jussieu et UVSQ)
# SECOURS=mailhost.jussieu.fr
SECOURS=mailhost.uvsq.fr

#       INTERPRETATION DU FICHIER XXX.CONFIG

#       Host= (defini)

#       Labo= (defini)

Domaine=$Labo.$SITE
ListeDomaines="$Labo.$SITE $Labo"

#       Mailhost=o/n

case "$Mailhost" in
```

```
# Si c'est le mailhost du labo

     o) Mailhost=LABO
        RelaisExterieur="mailhost.$SITE]:[$SECOURS"

#       SpoolCommun=o/n

        case "$SpoolCommun" in
             o) AdressesLocales=TOUT_DOMAINE
                AdressesInternes=RIEN
                ;;
             n) AdressesLocales=DOMAINE
                AdressesInternes=ETOILE_DOMAINE
                ;;
             *) echo "Variable SpoolCommun mal initialisee"
                exit 1
                ;;
        esac

#       SpoolMachines= (liste)

        case "$SpoolMachines" in
            "") ;;
             *) if [ "$SpoolCommun" = o ]
                then
                        AdressesLocales=LISTE
                        AdressesInternes=ETOILE_DOMAINE
                        ListeAdressesLocales="$SpoolMachines"
                else
                        echo "Incoherence entre SpoolCommun et SpoolMachine"
                        exit 1
                fi
                ;;
        esac

#       FromLabo=o/n

        case "$FromLabo" in
             o) ReecritureAdressesLocales=$Domaine
                ReecritureAdressesInternes=$Domaine
                ;;
             n) ReecritureAdressesLocales=$Host.$Domaine
                ReecritureAdressesInternes=
                ;;
             *) echo "Variable Fromlabo mal initialisee"
                exit 1
                ;;
        esac

#       FromLaboMachines= (liste)

#       Cette liste est la pour des raisons historiques.
#       En realite, si elle est non-vide, son contenu est
#       suppose etre le meme que celui de SpoolMachines.

        case "$FromLaboMachines" in
            "") ;;
             *) if [ "$FromLabo" = o ]
                then
```

```
                        ReecritureAdressesInternes=
                else
                        echo "Incoherence entre FromLabo et FromLaboMachines"
                        exit 1
                fi
                ;;
        esac

        ;;

# Si c'est une feuille du labo

     n) Mailhost=FEUILLE
        AdressesInternes=RIEN
        RelaisExterieur="mailhost.$Domaine"

#       ToutEnvoyer

        case "$ToutEnvoyer" in
             o) AdressesLocales=RIEN
                ;;
             n) AdressesLocales=HOST
                ;;
             *) echo "Variable ToutEnvoyer mal initialisee"
                exit 1
                ;;
        esac

        ;;

# Si c'est du genre inconnu

    *) echo "Variable Mailhost mal initialisee"
       exit 1
       ;;
esac

#       Variables "systemes"

#       Aliases

#       SendmailSt

RelaisExterieur="smtp.[$RelaisExterieur]"

#       RevAliases= (nom)

# On pourrait ici faire une verification de l'existence du dit fichier mais
#       - il peut ne pas encore exister au moment de la configuration
#       - il peut exister sur la machine destinatrice de ce sendmail.cf.
```

Configuration du mailhost du campus

Étant donné que le *mailhost* du campus est sous la responsabilité de l'administrateur réseau du campus, il n'y a pas besoin de traiter ce cas dans les exemples donnés aux administrateurs des laboratoires. On peut donc procéder à une configuration isolée du *mailhost* du campus, en utilisant directement les variables du configurateur.

Voici donc la configuration de `soleil.uvsq.fr` :

```
Host='soleil'
Domaine='uvsq.fr'
ListeDomaines="$Domaine"
AdressesLocales='DOMAINE'
AdressesInternes='ETOILE_DOMAINE'
ReecritureAdressesLocales='uvsq.fr'
MailhostEnInterne='o'
RelaisExterieur=''
MailerLocal='/usr/libexec/mail.local lsDFMAw5:/|@qrmn mail $u'
Aliases='/etc/aliases'
SendmailSt='/var/log/sendmail.st'
SendmailHf='/usr/share/misc/sendmail.hf'
Mqueue='/var/spool/mqueue'
RevAliases='hash -N /local/etc/revaliases'
TableRoutages='hash -N /local/etc/routages'
MailerUucp='/usr/bin/uux DFMhmsu uux - -r $h!rmail ($u)'
ListeNoire='hash -N /local/etc/liste-noire'
```

Ce fichier, sans commentaire, est particulièrement simple.

3.7.3 Institut Blaise Pascal

Avertissement

L'Institut Blaise Pascal a cessé d'exister le premier janvier 1997. À cette date, une partie des équipes a formé un nouveau laboratoire, le LIP6 (Laboratoire d'Informatique de Paris VI). Néanmoins, l'IBP continue de figurer dans les exemples car son architecture de distribution des courriers est très intéressante.

Architecture

Le cas de l'Institut Blaise Pascal est très intéressant, car le routage du courrier est très hiérarchisé dans peu de champs du DNS. Grossièrement, le domaine `ibp.fr` contient 3 champs[13], c'est-à-dire contient des noms de la forme *machine*`.ibp.fr`. L'IBP étant une fédération de laboratoires, il y a une machine par laboratoire (par exemple : `masi.ibp.fr`) vers laquelle transitent tous les courriers pour le laboratoire. Sur cette machine réside une base d'aliases et de revaliases qui permet de re-router les courriers vers des *mailhosts* d'équipes. Chaque *mailhost* d'équipe partage son répertoire de boîtes aux lettres avec ses feuilles.

En émission, lorsqu'un courrier part d'une feuille, il est transmis à son *mailhost* d'équipe qui le signe (avec une signature comme *login*`@`*equipe*`.ibp.fr`) puis ce courrier part vers le *mailhost* du laboratoire qui change la signature pour donner : *Prénom*`.`*Nom*`@`*laboratoire*`.ibp.fr`. Le courrier est ensuite renvoyé à `ibp.ibp.fr` qui réécrit éventuellement les adresses de certaines personnes des laboratoires en adresses dans le domaine `ibp.fr`.

[13] Sauf quelques sous-domaines, cas particuliers.

Fichiers de configuration

Dans cette architecture, on a donc 4 niveaux :

- le *mailhost* de l'IBP
- le *mailhost* d'un laboratoire (c'est en fait un *mailhost* d'équipe qui possède un deuxième nom dans le DNS)
- le *mailhost* d'équipe
- la feuille

Fichiers de configuration

La machine `ibp.ibp.fr` est le *mailhost* de l'IBP :

```
Host='ibp'
Mailhost='DOMAINE'
TableRoutages='hash -N /usr/local/etc/routages'
RevAliases='hash -N /usr/local/etc/revaliases'
Aliases='/etc/aliases'
SendmailSt='/etc/sendmail.st'
SendmailHf='/usr/lib/sendmail.hf'
Mqueue='/usr/spool/mqueue'
MailerLocal='/bin/mail DFMPlmnrs mail -d $u'
```

La machine `masi.ibp.fr` est le *mailhost* du laboratoire MASI :

```
Host='masi'
Mailhost='LABO'
Laboratoire='masi'
RevAliases='hash -N /usr/local/etc/revaliases'
ListeFeuilles='hermes eole'
Aliases='/etc/aliases'
SendmailSt='/etc/sendmail.st'
SendmailHf='/usr/lib/sendmail.hf'
Mqueue='/usr/spool/mqueue'
MailerLocal='/bin/mail DFMPlmnrs mail -d $u'
```

La machine `do.ibp.fr` est le *mailhost* d'une équipe :

```
Host='do'
Mailhost='EQUIPE'
ListeFeuilles='/usr/local/etc/feuilles'
Laboratoire='masi'

Aliases='/etc/aliases'
SendmailSt='/etc/sendmail.st'
SendmailHf='/usr/lib/sendmail.hf'
Mqueue='/usr/spool/mqueue'
MailerLocal='/bin/mail DFMPlmnrs mail -d $u'
```

La machine `la.ibp.fr` est une feuille :

```
Host='la'
Mailhost='FEUILLE'
Equipe='do'

Aliases='/etc/aliases'
SendmailSt='/etc/sendmail.st'
SendmailHf='/usr/lib/sendmail.hf'
Mqueue='/usr/spool/mqueue'
MailerLocal='/bin/mail DFMPlmnrs mail -d $u'
```

La machine `eole.ibp.fr` est une feuille dont le *mailhost* (`hermes.ibp.fr`) est en même temps le *mailhost* du laboratoire (`masi.ibp.fr`) :

```
Host='eole'
Mailhost='FEUILLE'
Equipe='masi'

Aliases='/etc/aliases'
SendmailSt='/etc/sendmail.st'
SendmailHf='/usr/lib/sendmail.hf'
Mqueue='/usr/spool/mqueue'
MailerLocal='/bin/mail DFMPlmnrs mail -d $u'
```

Script de traduction

```
#
# Regles correspondant au site ibp.fr
#
# Un mailhost de domaine        ibp.ibp.fr
# n mailhosts de laboratoire    labo.ibp.fr
# n*n mailhosts d'equipes       equipe.ibp.fr
# feuilles d'equipes            machine.ibp.fr
# Les feuilles d'equipes partagent un spool commun.
# Des sous-domaines gerant un mailhost  mailhost.sous-domaine.ibp.fr
#
# Signature homogene : Prenom.Nom@labo.ibp.fr faite au niveau
# des mailhosts de labo.
# Signature exceptionnelle : Prenom.Nom@ibp.fr faite au niveau
# du mailhost de l'ibp.
#
# Les courriers depuis l'exterieur passent par ibp, puis sont retransmis
# vers le mailhost de laboratoire qui les reroute via /etc/aliases
# vers le mailhost d'equipe concerne.
# Tout courrier provenant d'une machine d'equipe (feuille ou mailhost)
# est envoye au mailhost de laboratoire pour reecriture.
#
# Les courriers a destination des sous-domaines sont geres par routage
# explicite.
#

#
# Le nom du site. Utile pour les autres organismes qui veulent
# s'inspirer de ce fichier.
#
```

```
# SITE=votre-domaine.fr
SITE=ibp.fr

#
# Variables disponibles pour les fichiers de config (ceux
# fournis aux utilisateurs)
#
#       Host=
#       Mailhost=DOMAINE/LABO/EQUIPE/FEUILLE
#
#       Variables pour Mailhost=DOMAINE
#               TableRoutages= (fichier)
#               RevAliases= (fichier)
#
#       Variables pour Mailhost=LABO
#               RevAliases= (fichier)
#               ListeFeuilles= (liste machines)
#               Laboratoire= (labo)
#
#       Variables pour Mailhost=EQUIPE
#               ListeFeuilles= (liste machines)
#               Laboratoire= (labo)
#
#       Variables pour Mailhost=FEUILLE
#               Equipe= (machine)
#
#       + les variables classiques "systemes" (Aliases, etc.)
#

case "$Mailhost" in
    DOMAINE)
        Domaine=$SITE
        ListeDomaines="$Domaine"
        AdressesLocales=DOMAINE         # seulement pour ibp.fr
        AdressesInternes=ETOILE_DOMAINE
        RelaisExterieur=
        MailhostEnInterne=
        ReecritureAdressesLocales=$Domaine
        ReecritureAdressesInternes=
        ;;
    LABO)
        Domaine=$SITE
        ListeDomaines="$Domaine"
        AdressesLocales=LISTE
        AdressesInternes=ETOILE_DOMAINE
        ListeAdressesLocalesOuInternes="$ListeFeuilles"
        RelaisExterieur="smtp.[mailhost.$Domaine]"
        MailhostEnInterne=
        TableRoutages=
        ReecritureAdressesLocales=$Laboratoire.$Domaine
        ReecritureAdressesInternes=
        ;;
    EQUIPE)
        Domaine=$SITE
        ListeDomaines="$Domaine"
        AdressesLocales=LISTE
        AdressesInternes=RIEN
        ListeAdressesLocalesOuInternes="$ListeFeuilles"
        RelaisExterieur="smtp.[$Laboratoire.$Domaine]"
```

```
            MailhostEnInterne=
            TableRoutages=
            ReecritureAdressesLocales=$Host.$Domaine
            ReecritureAdressesInternes=
            ;;
        FEUILLE)
            Domaine=$SITE
            ListeDomaines="$Domaine"
            AdressesLocales=RIEN
            AdressesInternes=RIEN
            ListeAdressesLocalesOuInternes=
            RelaisExterieur="smtp.[$Equipe.$Domaine]"
            MailhostEnInterne=
            TableRoutages=
            ReecritureAdressesLocales=
            ReecritureAdressesInternes=
            ;;
        *)
            echo "Variable Mailhost mal definie" >&2
            exit 1
            ;;
    esac
```

3.7.4 Maison

Architecture

Cet exemple est celui d'un site individuel, c'est-à-dire d'un utilisateur isolé avec un pécé, muni d'un système d'exploitation digne de ce nom, relié via modem par l'intermédiaire d'un fournisseur de connectivité IP, que nous appellerons `fai` dans la suite.

En réception, la plupart des fournisseurs acceptent les adresses comme `nom@fai.fr`, et proposent ensuite les messages via un protocole comme POP ou IMAP (voir 1.9, page 32). On utilisera donc un outil comme `gwpop` ou `fetchmail`, lisant par exemple sur la machine `pop.fai.fr`.

De plus, les fournisseurs ne faisant généralement pas de la lutte anti-spam , on souhaite utiliser les listes noires de type RBL (voir 1.10.2, page 36) sur les courriers entrants.

En émission, on suppose que le fournisseur accepte les courriers via SMTP, sur la machine `smtp.fai.fr` (qui peut être la même machine).

Une configuration similaire (mais non identique) est décrite dans le Guide du Rootard Linux[14].

Configuration de fetchmail

Dans notre exemple, le programme `fetchmail` est utilisé pour récupérer le courrier (en utilisant POP ou IMAP). D'autre part, une fois le courrier récupéré, `fetchmail` sous-traite le dépôt dans la boîte aux lettres locale au serveur SMTP local, ce qui suppose bien évidemment que `sendmail` soit configuré.

[14] http://www.freenix.fr/linux/Guide/

Lorsque le système démarre, il faut lancer `fetchmail`, en mode *démon*, pour le compte de l'utilisateur qui doit recevoir le courrier. Par exemple :

```
su moi -c fetchmail -d 120 -t 20
```

Cette configuration utilise le fichier `~moi/.fetchmailrc`, qui pourrait être :

```
poll pop.fai.fr \\
      protocol pop3 \\
      username nom \\
      password toto
```

Le protocole spécifié est `pop3`, le plus courant. Le nom d'utilisateur chez le fournisseur (`nom`) ne correspond pas forcément au nom d'utilisateur sur la machine locale (`moi`). Enfin, le mot de passe est fourni en clair dans ce fichier.

Configuration de sendmail

Dans cet exemple, il n'est pas question de faire un fichier de règles comme décrit en 3.4.2 (page 63). On utilisera donc un fichier de règles vide.

Le fichier de variables est particulièrement simple :

```
Host='monpc'
Domaine='fai.fr'
ListeDomaines="$Domaine"
AdressesLocales='HOST'
AdressesInternes='RIEN'
ReecritureAdressesLocales=$Domaine
RelaisExterieur="smtp.[mail.$Domaine]"
MailerLocal='/usr/libexec/mail.local lsDFMAw5:/|@qrmn mail $u'
Aliases='/etc/mail/aliases'
SendmailSt='/var/log/sendmail.st'
SendmailHf='/etc/mail/helpfile'
Mqueue='/var/spool/mqueue'
FichierInclude='vide/include-monpc.cf'
ListeNoire='hash -N /local/etc/listenoire'
```

On notera en particulier la définition de la variable `ListeNoire`, bien que `RelaisExterieur` soit défini. Ceci a pour effet de définir les règles de filtrage. Leur utilisation a lieu dans le fichier inclus :

```
_###############################################################################
_#                              MON FILTRAGE PERSONNEL
_###############################################################################

_#
_# N'étant pas forcément maître des relais de messagerie, je filtre
_# sur le champ Received. Pour cela, j'intercepte tous les Received:
_# et je traite chaque contenu en totalité, sans considérer que les
_# parenthèses introduisent un commentaire (d'où le "+").
```

```
_#
HReceived:      $>+check_received

_#
_# La classe des relais de messagerie à partir desquels je vérifie
_# le client SMTP.
_#
CR mail1.fai.fr mail2.fai.fr mail3.fai.fr

_#
_# Filtrage du champ "Received". Je suppose que mes relais décrits
_# dans la classe R ajoutent ce champ, structuré de la façon suivante :
_#   Received: ...from... (...[<adr IP du client SMTP>]...) by <relais>...
_#
Scheck_received
R from $* [$-.$-.$-.$-] $* by $=R $*    $: $>error_relay_addr $2.$3.$4.$5

Serror_relay_addr
R $*                    $: $>relay_addr $1
# R ERREUR . $*         $#discard $: $1                 un peu violent...
R ERREUR . $*           $#error $: $1
```

Enfin, le programme `sendmail` est lancé avec les arguments :

```
/usr/sbin/sendmail -bd -q2m
```

L'option `-bd` est nécessaire pour que `fetchmail` puisse déposer le courrier reçu. L'option `-q2m` indique, quant à elle, que `sendmail` doit vérifier la file d'attente toutes les deux minutes. Dans le cas d'une connexion au fournisseur, il s'agit d'un intervalle généralement suffisant pour que les courriers puissent partir.

3.7.5 Utilisation de LDAP

Cette configuration n'a comme seul intérêt que de montrer un exemple d'utilisation de LDAP pour obtenir toutes les informations possibles, à savoir les aliases, les revaliases, la liste noire et la table de routage.

On notera que les variables du kit devant toutes tenir sur une seule ligne, les informations relatives à LDAP ne sont pas très faciles à lire.

```
Host='machine'
Domaine='domaine.fr'
ListeDomaines="$Domaine"
AdressesLocales='HOST'
AdressesInternes='RIEN'
ReecritureAdressesLocales=$Host.$Domaine
MailerLocal='/usr/libexec/mail.local lsDFMAw5:/|@qrmn mail $u'
SendmailSt='/etc/mail/statistics'
SendmailHf='/etc/mail/helpfile'
Mqueue='/var/spool/mqueue'
#
# Tout ce qui suit concerne LDAP
#
ParametresLDAP='-h ldap.domaine.fr -p 389 -b dc=domaine,dc=fr'
```

```
Aliases='ldap:'
#
# les trois utilisations de LDAP ci-dessous, pour des raisons obscures
# de comportement du Shell, doivent être à chaque fois sur une seule ligne.
#
RevAliases='ldap -v sendmailMTAAliasValue -k "(&(objectClass=sendmailMTAAliasObject) (sendmailMTAAliasGrouping=revaliases)(|(sendmailMTACluser=${sendmailMTACluster})(sendmailMTAHost=$j))(sendmailMTAKey=%0))"'
ListeNoire='ldap -v sendmailMTAMapValue -1 -k "(&(objectClass=sendmailMTAMapObject)(sendmailMTAMapName=listenoire)(|(sendmailMTACluser=${sendmailMTACluster})(sendmailMTAHost=$j))(sendmailMTAKey=%0))"'
TableRoutage='ldap -v sendmailMTAMapValue -1 -k "(&(objectClass=sendmailMTAMapObject)(sendmailMTAMapName=routages)(|(sendmailMTACluser=${sendmailMTACluster})(sendmailMTAHost=$j))(sendmailMTAKey=%0))"'
```

3.8 Implémentation

Cette section décrit succintement le fonctionnement du script de configuration.

3.8.1 Le script shell

Le configurateur est un simple *script shell* fonctionnant suivant les principes suivants :

1. lecture du fichier de configuration de l'utilisateur avec la commande « `.` » (point) du Shell de Bourne pour accéder aux variables ;
2. mise en forme de ces variables ;
3. utilisation d'un « *here document* » (c'est-à-dire redirection immédiate avec des données contenues dans le script lui-même) pour inclure un `sendmail.cf` modèle et appel du préprocesseur du langage C (`cpp`) pour sélectionner (`#ifdef`) ou paramétrer (`#define`) certaines parties du modèle suivant les options choisies par l'administrateur ;
4. le fichier généré est envoyé sur la sortie standard.

Ce configurateur est très portable : il a été testé sur de nombreuses architectures. Même s'il ne fonctionnait pas sur un type de système, ce ne serait pas gênant : une de ses caractéristiques est que le fichier généré est indépendant de la machine qui l'a généré. Par exemple, il est possible d'exécuter le configurateur sur un HP à l'UVSQ pour générer un `sendmail.cf` pour un Sun de Jussieu.

Le choix du préprocesseur du langage C s'est imposé pour plusieurs raisons :

- la toute première raison est que les auteurs ont une bonne habitude de ce programme ;
- il est relativement portable, même s'il y a quelques différences entre le langage C traditionnel et le langage C tel que normalisé par l'ANSI ;
- la syntaxe est très simple (`#ifdef` par exemple) et peu sujette à des traquenards, comme pour `m4` et ses `dnl` ;
- on trouve ce programme sur presque tous les systèmes.

3.8.2 Le fichier généré

Le fichier généré est un `sendmail.cf` relativement classique, à quelques exceptions près, décrites ci-après.

Canonisation des adresses

Comme indiqué dans le chapitre précédent (voir 2.5.4, page 48), la forme canonique des adresses est la forme classique suffixée par une classe :

> *partie-locale* <`@` *premier-relais* `.` *classe* > *routage*

La *classe* est :

- `LOCAL`, auquel cas l'adresse est considérée comme provenant de (si elle figure dans un champ `From` ou assimilé) ou destinée à (si elle figure dans un champ `To` ou assimilé) la machine locale ;
- `INTERNE`, auquel cas l'adresse est considérée comme faisant partie de la juridiction de cette machine (qui est vraisemblablement un *mailhost*), mais sans être locale ;
- `EXTERNE`, dans tous les autres cas.

Cette classification est assez délicate, mais elle est concentrée dans un seul ensemble de règles (l'ensemble 19).

Réécriture des adresses numériques

Les adresses numériques (adresses IP) sont traitées de manière particulière : toutes les adresses numériques sont réécrites en adresses symboliques si c'est possible.

Une adresse numérique de destinataire (dans l'enveloppe) n'est acceptée que si l'adresse correspondante peut être traduite en adresse symbolique. Ce comportement n'est pas tout à fait conforme à la RFC 1123, mais cela a permis l'extension du DNS sur le campus de Jussieu.

Ensembles de règles

Les ensembles de règles définis dans le fichier généré sont :

- 0
  L'ensemble classique pour router le courrier à partir de l'adresse du destinataire dans l'enveloppe. Cet ensemble consulte la table des routages explicites si celle-ci est définie (variable `TableRoutages`) ;
- 1
  Les réécritures des adresses (`ReecritureAdressesLocales` et `ReecritureAdressesInternes`), ainsi que la transformation des noms de login en *Prenom.Nom* si celle-ci est sélectionnée (par le biais de la variable `RevAliases`) sont effectuées dans cette règle, ce qui permet d'éviter de le faire pour chaque agent de transport ;
- 2
  Cet ensemble est vide, aucune transformation n'est effectuée sur l'adresse du destinataire ;
- 3
  La canonisation est effectuée dans cet ensemble. Tout cet ensemble est classique, sauf l'appel à la règle 19 qui fait la classification des adresses en trois catégories ;
- 4
  C'est la décanonisation. Le fonctionnement est minimal et très simple ;
- 16
  Cet ensemble est appelé par les ensembles spécialisés pour chaque agent de transport. Lorsqu'on communique

une adresse locale (non qualifiée) à un site extérieur, il faut transmettre une adresse qualifiée correctement. C'est le but de cet ensemble ;
- 17
  L'ensemble 17 réalise les réécritures décrites plus haut dans l'ensemble 1. En fait, l'ensemble 1 se contente d'appeler l'ensemble 17 ;
- 18
  Cet ensemble a pour but de rappeler récursivement les ensembles 3 puis 0 : lorsqu'une adresse avec routage explicite est présentée, et que le premier relais correspond à notre machine, on enlève la première partie et on recommence l'analyse sans ouvrir de connexion inutile ;
- 19
  La classification des adresses est effectuée dans cet ensemble ; en sortie, toute adresse est suffixée par `LOCAL`, `INTERNE` ou `EXTERNE` ;
- 10 et 20
  Ces deux ensembles effectuent les réécritures spécialisées pour l'agent de remise physique (agents `local` et `prog`) ;
- 11 et 21
  Ces deux ensembles effectuent les réécritures spécialisées pour l'agent de transport SMTP (agents `smtp` et `buggysmtp`) ;
- 12 et 22
  Ces deux ensembles effectuent les réécritures spécialisées pour l'agent de transport UUCP (agent `uucp`).
- `check_relay`
  Aucune vérification n'est effectuée dans cet ensemble, car le code d'erreur renvoyé par la version 8.8 de `sendmail` est toujours `550 Access denied`. On préfère placer ces vérifications dans la règle `check_mail` afin de fournir des codes d'erreurs plus explicites.
- `check_mail`
  Le courrier est rejeté si l'adresse n'est pas qualifiée et si le client SMTP est externe, si l'adresse provient d'un domaine cité dans la liste noire, ou si l'adresse est interne et que le client est externe (sauf s'il figure explicitement dans les exceptions de la liste noire).
  De plus, on effectue les vérifications qui devraient logiquement figurer dans `check_relay` : le client SMTP doit être enregistré dans le DNS (sauf si la ligne `NUMERIQUE OK` figure dans le fichier `ListeNoire`), et son adresse IP et son nom ne doivent figurer ni dans la liste noire locale, ni dans une des listes de type RBL (si la ligne `RBL SPAM ...` figure dans le fichier `ListeNoire`).
- `check_rcpt`
  Le courrier est rejeté si l'adresse de destination est externe et que le client SMTP est lui-même externe, sauf s'il figure dans les exceptions de la liste noire.
- `check_compat`
  Le courrier est rejeté si les deux adresses sont externes, sauf si l'une des deux figure dans les exceptions de la liste noire.

De plus, il existe d'autres ensembles de règles mineurs. Le lecteur intéressé est invité à se référer au script `configurateur` (largement commenté) pour plus de détails.

3.9 Test d'une configuration

La méthode générale pour tester un fichier de configuration est expliquée en détail dans le chapitre précédent (voir section 2.7, page 52). Cette section explicite quelques tests spécifiques aux fichiers de configuration générés par le kit de Jussieu.

3.9.1 Quelques erreurs à ne pas commettre

Tout d'abord, il faut savoir qu'une erreur classique consiste à ne pas orthographier correctement une variable. Pour certaines, cela a un effet désastreux : comme la variable n'est pas bien orthographiée, le Shell considère qu'elle n'existe pas. S'il s'agit d'une variable comme `RelaisExterieur`, par exemple, cela signifie qu'elle sera considérée comme non définie, d'où vraisemblablement un fichier généré qui n'aurait pas l'effet escompté.

Une autre erreur qui peut se produire consiste, comme nous l'avons vu en 3.6 (page 83), à utiliser un nom de machine ressemblant à un nom de constructeur, qui peut être mal interprété par le préprocesseur du langage C.

Enfin, pour vous simplifier la vie et éviter des erreurs dûes à l'expansion du `$` par le Shell, mettez le contenu de toutes les variables entre apostrophes simples (caractère `'`).

3.9.2 Tests des règles de réécriture

Étant donné que la configuration découpe les adresses en trois catégories, à savoir les adresses locales, internes, et extérieures (voir 3.5.2, page 67), plus les exceptions (avec la variable `TableRoutages`), la première chose à tester est ce découpage. Pour cela, il faut choisir quelques adresses représentatives dans chaque catégorie, et tester les règles avec l'option `-bt` (voir 2.7.2, page 54).

Les catégories sont définies par vos variables de configuration. Par exemple, si `AdressesLocales='RIEN'`, il est clair que la catégorie des adresses locales est vide et qu'il n'y a aucun test à faire. Si à l'autre extrême, `AdressesLocales='TOUT_DOMAINE'` et `Domaine='x.fr'` par exemple, toute adresse de `x.fr` est locale, donc il faudra tester, par exemple, les adresses `moi`, `moi@machine.x.fr`, `moi@x.fr`, etc.

Vérification du mailer et du relais

Le premier test consiste à vérifier si la classification effectuée par le fichier `sendmail.cf` correspond bien à vos attentes. Pour cela, le plus simple est de tester à la fois la règle 19 (classification) et la règle 0 (sélection du *mailer*). Il faut donc appliquer les règles 3 et 0 pour les adresses de chaque type :

– adresses locales
  Vous devez voir le suffixe `.LOCAL` ajouté en sortie de la règle 3, puis le *mailer* `local` sélectionné en sortie de la règle 0. L'argument après le `$:` doit correspondre au nom d'utilisateur figurant dans l'adresse. Si vous avez saisi un alias (par exemple `Pierre.David@prism.uvsq.fr`), vous devez voir le nom original (`Pierre.David`) après le `$:` car l'expansion des aliases n'est pas encore effectuée à ce niveau (elle le sera avant la remise finale). À ce stade, il n'y a pas non plus de vérification de validité de l'utilisateur.
– adresses internes
  Si vous avez des adresses internes (par exemple `moi@a.b.fr`), vous devez voir le suffixe `.INTERNE` ajouté en sortie de la règle 3, puis le *mailer* `smtp` sélectionné en sortie de la règle 0. L'argument après le `$@` doit correspondre à `a.b.fr` ou `mailhost.a.b.fr` (suivant le contenu de la variable `MailhostEnInterne`). Enfin, cet argument doit être encadré par des crochets pour indiquer qu'il ne faut pas utiliser les MX.
– adresses extérieures
  Vous devez voir le suffixe `.EXTERNE` ajouté en sortie de la règle 3, puis le *mailer* `smtp` sélectionné en sortie de la règle 0. L'argument après le `$@` est soit le nom du relais entre crochets (si vous avez positionné la variable `RelaisExterieur`), soit le nom de la machine concernée (sans crochets).
– adresses exceptionnelles
  Si vous avez affecté la variable `TableRoutages`, vous devez avoir saisi et compilé le fichier correspondant avant

de procéder aux tests. Essayez alors quelques adresses, ne tenez pas compte du suffixe ajouté lors de la règle 3, mais vérifiez soigneusement que la règle 0 conduit bien à ce que vous avez spécifié dans le fichier.

Vérification des réécritures d'adresses

Les règles de réécriture des adresses varient suivant le *mailer* sélectionné par la règle 0. Pour chaque *mailer*, il faut vérifier l'adresse d'expéditeur (qui peut être de l'une des trois classes, plus les exceptions, pour chacun des *mailers*) et l'adresse destinataire (qui doit correspondre aux classes conduisant à ce *mailer*).

Le tableau suivant donne les numéros des règles à tester pour chaque *mailer*. Par exemple, un courrier qui est remis via le *mailer* `smtp` fait que `sendmail` passe ses adresses d'expéditeur (champs `From:`, `Reply-To:`, etc) par les règles 3, 1, 10 et enfin 4. Pour plus de détails, voir la figure 2.3 (page 45).

Mailer	**Expéditeur**	**Destinataire**
`local`	3,1,10,4	3,2,20,4
`smtp`	3,1,11,4	3,2,21,4
`uucp`	3,1,12,4	3,2,22,4

Note : le *mailer* `uucp` n'est utile que dans le cas où vous avez défini `TableRoutages` et que vous y faites explicitement référence à `uucp`.

Si vous avez positionné la variable `RevAliases`, il faut vérifier que l'expansion des aliases inverses fonctionne correctement, c'est-à-dire qu'un nom d'expéditeur (et seulement d'expéditeur) est traduit, à l'issue de la règle 17, en fonction de ce fichier.

Vérification des règles anti-spam

Si vous avez spécifié la variable `ListeNoire`, il peut être de bon ton de tester l'efficacité des règles anti-*spam* (voir 2.5.5, page 48).

Pour cela, il faut positionner deux macros essentielles de `sendmail` : `client_name` et `client_addr`, qui contiennent respectivement le nom (ou son adresse IP entre crochets s'il n'en a pas) et l'adresse IP du client SMTP. Par exemple :

```
# sendmail -bt
> .D{client_name}[1.2.3.4]
> .D{client_addr}1.2.3.4
```

ou

```
# sendmail -bt
> .D{client_name}soleil.uvsq.fr
> .D{client_addr}193.51.24.1
```

Une fois ces deux macros définies, il est possible de tester les règles suivantes :

– `check_rcpt`

Il faut spécifier l'adresse du destintaire. Par exemple :

```
> check_rcpt truc@ailleurs.fr
```

Pour cette règle, il faut tester les combinaisons suivantes :
- client SMTP extérieur (ou non autorisé dans la liste noire) et adresse de destinataire extérieure

– `check_mail`
Il faut spécifier l'adresse de l'expéditeur. Par exemple :

```
> check_mail machin@spammeur.com
```

Pour cette règle, il faut tester les combinaisons suivantes :
- client SMTP enregistré dans la liste noire
- client SMTP non enregistré dans le DNS
- adresse électronique enregistrée dans la liste noire
- client SMTP extérieur (et non autorisé dans la liste noire) et adresse d'expéditeur incomplète (sans `@`)

– `check_compat`
Cette règle réalise un sous-ensemble des tests effectués dans la règle `check_rcpt`. Il n'est donc pas besoin de la tester.

Pour chacun des cas cités ci-dessus, la règle doit sélectionner le *mailer* `error` avec un message approprié. Pour toute autre combinaison, la règle doit renvoyer n'importe quoi, ce qui signifie que le courrier sera accepté.

3.9.3 Suite des tests

L'étape suivante consiste à tester l'expédition via les *mailers*. Pour cela, pour chaque catégorie d'adresse, envoyez un courrier directement à l'aide de `sendmail` comme décrit en 2.7.4 (page 58) en n'oubliant pas l'option `-v`.

Si chaque message arrive bien à destination et que les en-têtes sont correctes, vous êtes presque au bout de vos peines : il reste à installer le nouveau fichier de configuration, (re)démarrer `sendmail` et à tester l'arrivée des messages locaux via SMTP.

Si vous êtes arrivés jusqu'ici et que cette dernière étape se passe bien, vous pouvez alors envoyer une bouteille de champagne aux auteurs du kit...

Annexe A

Historique des versions

Cette annexe décrit l'historique des versions de la présente documentation.

- Version 5.4 – novembre 2001
  - Adaptation à la version 5.4 du kit : changement de format des RBL dans le fichier référencé par `ListeNoire`.
  - Explication de la soumission de messages avec la version 8.12 de `sendmail`.
  - Ajout d'une méthode pour obtenir un fichier `submit.cf`.
  - Tous les noms de fichiers et de répertoires sont en principe maintenant répertoriés dans l'index.
  - Exemple de lutte anti-spam même si le relais n'en pratique pas, voir 3.7.4, page 116.
  - Explication de l'utilisation de LDAP avec `sendmail`
  - Explication de l'utilisation de LDAP dans le kit, notamment des variables `Aliases`, `RevAliases`, `ListeNoire` et `TableRoutages`
  - Ajout de la variable `ParametresLDAP`.
- Version 5.3.2 – juillet 1999
  - Adaptation à la version 5.3.2 du kit : variable `TailleMaximum`, modification de `ListeDomaines`, relations entre `TableRoutages` et `ListeNoire`.
  - Explications sur le temps de conservation des courriers dans la file d'attente
  - Suppression de quelques coquilles.
  - Ajout d'un début d'index, référençant les variables pour le moment.
- Version 5.3.1 – juillet 1998
  - Adaptation à la version 5.3.1 du kit : disparition de la description de la variable `V8`, et ajout de la description de la variable `URL`.
  - Suppression des annexes sur les nouveautés de la version 8 de `sendmail`, et sur la syntaxe du fichier de configuration. Ces annexes sont obsolètes, et malgré les appels répétés dans les versions précédentes de cette documentation, une seule personne a manifesté son intérêt pour les voir réactualisées.
  - Ajout des vérifications des permissions effectuées par la version 8.9 de `sendmail`.
  - Ajout de la méthode de test des règles anti-*spam*.
  - Suppression de quelques coquilles.
- Version 5.3 – avril 1998
  - Adaptation à la version 5.3 du kit : documentation des règles anti-*spam*, ajout de la description des variables `ListeNoire`, `CodeErreur`, `FichierInclude`, `MailerUucp` et `SansCanonisation`.
  - Effort de documentation du *spam* et des règles anti-*spam* de `sendmail`.
  - Ajout d'un paragraphe sur la centralisation du traitement des courriers.
  - Réactualisation des RFC, de quelques sous-types MIME (text/HTML et multipart/related), ainsi que de l'authentification ESMTP.
  - Ajout de l'exemple du site « maison », car il y a de plus en plus d'utilisation du kit par des individuels.

  - Nombreuses modifications et corrections de bugs.
  - Ajout de *barres de changement* dans la marge, pour signaler les nouveautés.
  - Première version accessible via le Web.
- Versions 5.2 et antérieures
  Leur histoire se perd dans la nuit des temps...

Bibliographie

[1] Frey D., Adams R. *!%@:: A Directory of Electronic Mail Addressing and Networks* O'Reilly 1993.

[2] Costales B., Allman E., Rickert N. *Sendmail* O'Reilly 1993.

[3] Comer D. *Internetworking with TCP/IP, 3rd edition* Prentice Hall 1995.

[4] Albitz P., Liu C. *DNS and BIND* O'Reilly 1992.

[5] Nemeth E., Snyder G., Seebass S. *Unix System Administration Handbook.* Prentice Hall 1989.

[6] David P., Thibault J. *L'intégration du réseau de Jussieu dans le réseau Internet.* Disponible sur demande auprès des auteurs.

[7] David P., Thibault J. *Réseau du Campus Jussieu / Domain Name System – Fonctionnement et Installation / Messagerie – Architecture et Installation* Disponible sur demande auprès des auteurs.

[8] Postel J.B. *Simple Mail Transfer Protocol* Request For Comments 821, 1982

[9] Crocker D.H. *Standard for the Format of ARPA Internet Text Messages* Request For Comments 822, 1982

[10] Stahl M. *Domain Administrators Guide* Request For Comments 1032, 1987

[11] Lottor M. *Domain Administrators Operations Guide* Request For Comments 1033, 1987

[12] Mockapetris P. *Domain Names - Concepts and Facilities* Request For Comments 1034, 1987

[13] Mockapetris P. *Domain Names - Implementation and Specification* Request For Comments 1035, 1987

[14] Mockapetris P. *DNS Encoding of Network Names and Other Types* Request For Comments 1101, 1989

[15] Borenstein N., *A User Agent Configuration Mechanism for Multimedia Mail Format Information* Request For Comments 1343, 1992

[16] Borenstein N. *Implications of MIME for Internet Mail Gateways* Request For Comments 1344, 1992

[17] Vaudreuil G. *Transition of Internet Mail from Just-Send-8 to 8bit-SMTP/MIME* Request For Comments 1428, 1993

[18] Borenstein N. *The text/enriched MIME Content-type* Request For Comments 1563, 1994

[19] Klensin J., Freed N., Rose M., Stefferud E., Crocker D. *SMTP Service Extension for 8bit-MIME Transport* Request For Comments 1652, 1994

[20] Vaudreuil G. *SMTP Service Extensions for Transmission of Large and Binary MIME Messages* Request For Comments 1830, 1995

[21] Crocker D., Freed N., Cargille A. *SMTP Service Extension for Checkpoint/Restart* Request For Comments 1845, 1995

[22] Klensin J., Freed N., Rose M., Stefferud E., Crocker D. *SMTP Service Extensions* Request For Comments 1869, 1995

[23] Klensin J., Freed N., Moore K. *SMTP Service Extension for Message Size Declaration* Request For Comments 1870, 1995

[24] Moore K. *SMTP Service Extension for Delivery Status Notifications* Request For Comments 1891, 1996

[25] Vaudreuil G. *The Multipart/Report Content Type for the Reporting of Mail System Administrative Messages* Request For Comments 1892, 1996

[26] Vaudreuil G. *Enhanced Mail System Status Codes* Request For Comments 1893, 1996

[27] Moore K., Vaudreuil G. *An Extensible Message Format for Delivery Status Notifications* Request For Comments 1894, 1996

[28] Rose M. *Post Office Protocol - version 3* Request For Comments 1939, 1996

[29] De Winter J. *SMTP Service Extension for Remote Message Queue Starting* Request For Comments 1985, 1996

[30] Freed N., Moore K., Cargille A. *Definition of the URL MIME External-Body Access-Type* Request For Comments 2017, 1996

[31] Borenstein N., Freed N. *MIME (Multipurpose Internet Mail Extensions) Part One: Format of Internet Message Bodies* Request For Comments 2045, 1996

[32] Borenstein N., Freed N. *MIME (Multipurpose Internet Mail Extensions) Part Two: Media Types* Request For Comments 2046, 1996

[33] Moore K. *Representation of Non-ASCII Text in Internet Message Headers MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text* Request For Comments 2047, 1996

[34] Freed N., Klensin J., Postel J. *MIME (Multipurpose Internet Mail Extensions) Part Four: Registration Procedures* Request For Comments 2048, 1996

[35] Freed N., Borenstein N. *MIME (Multipurpose Internet Mail Extensions) Part Five: Conformance Criteria and Examples* Request For Comments 2049, 1996

[36] Crispin M. *Internet Message Access Protocol - Version 4rev1* Request For Comments 2060, 1996

[37] Palme J. *Common Internet Message Headers* Request For Comments 2076, 1997

[38] Palme J., Hopmann A. *MIME E-mail Encapsulation of Aggregate Documents, such as HTML* Request For Comments 2110, 1997

[39] Levinson E. *The MIME Multipart/Related Content-type* Request For Comments 2112, 1997

[40] Crocker D. *Mailbox Names for Common Services, Roles and Functions* Request For Comments 2142, 1997

[41] Freed N. *SMTP Service Extension for Command Pipelining* Request For Comments 2197, 1997

[42] Myers J. *Simple Authentication and Security Layer (SASL)* Request For Comments 2222, 1997

[43] Hewlett-Packard *Installing and Administering ARPA services* Manuels de HP-UX 8.0, Février 1991

Index