solutions
Unix • Linux

THIBAUT **MAQUET**

Sendmail

Installer, administrer et optimiser un serveur de messagerie

Avec la contribution de Olivier Salvatori

ISBN 2-212-11262-9
EYROLLES

6

Administration de Sendmail

Une fois Sendmail installé, il est nécessaire de maîtriser les opérations élémentaires de démarrage et d'arrêt du programme. Pour ce faire, il existe deux méthodes : l'une classique, utilisée depuis le début de Sendmail, et l'autre plus évoluée et développée par des distributeurs Linux. Ces deux méthodes ont leurs avantages et leur inconvénients.

Dans ce chapitre, vous verrez que Sendmail dispose de très nombreux paramètres en ligne de commande qui permettent de modifier son comportement en mode opératoire. Vous verrez également comment il gère la file d'attente dans laquelle sont contenus les messages en instance de départ.

Démarrer et arrêter Sendmail

Comme expliqué précédemment, il existe deux façons de procéder au démarrage et à l'arrêt de Sendmail, selon que vous utilisez n'importe quelle plate-forme UNIX ou que vous travaillez sur un environnement compatible Redhat.

La première méthode, dite classique, est celle qui est utilisée depuis les débuts de Sendmail. Elle consiste à lancer l'exécutable Sendmail assorti de quelques paramètres. La seconde fait appel à des scripts SHELL qui simplifient grandement l'arrêt et le démarrage de Sendmail.

La méthode classique

Pour démarrer Sendmail selon la méthode classique, il faut le lancer en mode service et lui indiquer l'intervalle d'exécution de la gestion de la file d'attente. Cela se fait *via* les options `bd` et `q`.

Pour démarrer Sendmail, lancez la commande :

```
# /usr/sbin/sendmail -bd -q15m
```

Cela indique que Sendmail va fonctionner en mode détaché, s'installer sur le port 25 et gérer la file d'attente toutes les quinze minutes.

Vérifiez que le service fonctionne correctement au moyen de la commande suivante :

```
# ps -fax
18626 ?        S      1:09 sendmail: accepting connections
```

Vérifiez également que vous avez bien installé un programme sur le port 25 (SMTP) :

```
# netstat -a | grep :smtp
tcp        0      0 *:smtp                  *:*                     LISTEN
```

Vous allez maintenant essayer d'accéder à ce service *via* un simple Telnet :

```
# telnet 127.0.0.1 25
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
220 s1.tutu.com ESMTP Sendmail 8.11.5/8.11.5; Sat, 25 Aug 2001 14:49:16 +0200
```

Vous avez bien le code de retour `220`, qui indique une réussite d'une transaction SMTP et s'accompagne de l'en-tête de Sendmail.

Lors de l'exécution de Sendmail, celui-ci crée le fichier `/var/run/sendmail.pid` — l'emplacement de ce fichier peut différer suivant les plates-formes —, qui contient son identificateur de processus ainsi que la commande qui a permis son démarrage.

En voici un exemple :

```
9261
/usr/sbin/sendmail -bd -q15m
```

Lorsque vous lancez Sendmail, il est recommandé de spécifier le chemin d'accès vers l'exécutable. Si vous omettez le répertoire, Sendmail envoie le message suivant :

```
daemon invoked without full pathname; kill -1 won't work
```

Lorsque vous faites relire à Sendmail sa configuration *via* un signal `HUP` (voir plus loin), Sendmail utilise la deuxième ligne du fichier `sendmail.pid`. Sans l'indication du chemin d'accès, il se trouve dans l'impossibilité de trouver son exécutable.

Pour arrêter Sendmail, il suffit de lui lancer un signal `TERM` au moyen de l'instruction `kill`. Le signal se fait sur le service écoutant sur le port 25, celui qui porte le commentaire *sendmail : accepting connections*. Les éventuelles autres instances de Sendmail sont celles qui sont en train de gérer la file d'attente ou d'accepter des messages en entrée.

Pour stopper Sendmail, lancez :

```
# kill PID de Sendmail
```

ou :

```
# kill `head -1 /var/run/sendmail.pid`
```

Un signal TERM est normalement sans danger pour Sendmail. En revanche, il vaut mieux éviter d'envoyer un signal KILL (kill -9) sous peine de perdre des messages en instance.

Il peut être nécessaire de faire relire sa configuration à Sendmail, notamment lorsque vous modifiez le fichier sendmail.cf, *via* le fichier sendmail.mc, ou que vous ajoutez des domaines à gérer dans le fichier local-host-names (ou sendmail.cw). Dans ce cas, il vous suffit d'envoyer un signal HUP à Sendmail pour que ce dernier enclenche la relecture de la configuration :

```
# kill -HUP PID de Sendmail
```

ou :

```
# kill -HUP `head -1 /var/run/sendmail.pid`
```

Dans le fichier d'historique, la relecture de la configuration se caractérise par les événements suivants :

```
May  9 12:44:26 mail sendmail[4411]: restarting /usr/sbin/sendmail due to signal
May  9 12:44:30 mail sendmail[9203]: starting daemon (8.11.6): SMTP+queueing@01:00:00
```

La méthode évoluée

Certains éditeurs de distribution Linux, notamment Redhat, simplifient le mécanisme d'exécution de Sendmail. Il devient alors inutile d'aller chercher l'identificateur de Sendmail. L'arrêt et le démarrage du logiciel, mais aussi la création des bases de données, se font très facilement *via* une commande unique.

L'avantage de cette méthode est évident pour le néophyte, puisque les manipulations complexes de Sendmail sont limitées au strict minimum.

Cette commande fonctionne de la façon suivante :

```
service sendmail commande
```

où commande peut être :

- start pour le démarrage, la lecture de la configuration et la création des bases de données ;
- stop pour l'arrêt ;
- restart ou reload pour le redémarrage ;
- status pour l'état du service.

Tout autre commande affiche la façon d'utiliser le script :

```
Usage: sendmail {start|stop|restart|status}
```

Le script suivant simplifie grandement la manipulation de Sendmail en proposant une succession de tests visant à garantir l'intégrité du service :

```
#!/bin/sh
#
# sendmail      This shell script takes care of starting and stopping
#               sendmail.
#
# chkconfig: 2345 80 30
# description: Sendmail is a Mail Transport Agent, which is the program \
#              that moves mail from one machine to another.
# processname: sendmail
# config: /etc/sendmail.cf
# pidfile: /var/run/sendmail.pid

# Source function library.
. /etc/rc.d/init.d/functions

# Source networking configuration.
. /etc/sysconfig/network

# Source sendmail configureation.
if [ -f /etc/sysconfig/sendmail ] ; then
        . /etc/sysconfig/sendmail
else
        DAEMON=yes
        QUEUE=1h
fi

# Check that networking is up.
[ ${NETWORKING} = "no" ] && exit 0

[ -f /usr/sbin/sendmail ] || exit 0

RETVAL=0

# See how we were called.
case "$1" in
  start)
        # Start daemons.

        echo -n "Starting sendmail: "
        /usr/bin/newaliases > /dev/null 2>&1
        for i in virtusertable access domaintable mailertable ; do
            if [ -f /etc/mail/$i ] ; then
                makemap hash /etc/mail/$i < /etc/mail/$i
            fi
        done
        daemon /usr/sbin/sendmail $([ "$DAEMON" = yes ] && echo -bd) \
                                  $([ -n "$QUEUE" ] && echo -q$QUEUE)
        RETVAL=$?
        echo
        [ $RETVAL -eq 0 ] && touch /var/lock/subsys/sendmail
        ;;
```

```
  stop)
        # Stop daemons.
        echo -n "Shutting down sendmail: "
        killproc sendmail
        RETVAL=$?
        echo
        [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/sendmail
        ;;
  restart|reload)
        $0 stop
        $0 start
        RETVAL=$?
        ;;
  status)
        status sendmail
        RETVAL=$?
        ;;
  *)
        echo "Usage: sendmail {start|stop|restart|status}"
        exit 1
esac

exit $RETVAL
```

Nous allons détailler le fonctionnement de ce script ligne à ligne.

La ligne suivante contient les fonctions primitives de manipulation de processus en shell (`killproc`, `daemon`, etc.) :

```
. /etc/rc.d/init.d/functions
```

Voici les variables d'environnement réseau :

```
. /etc/sysconfig/network
```

Dans l'extrait suivant, si le fichier `/etc/sysconfig/sendmail` existe, il est exécuté. Ce fichier contient le mode d'exécution du service, ainsi que l'intervalle d'exécution de la gestion de la file d'attente. Ces paramètres sont positionnés *via* deux variables d'environnement, `DAEMON` et `QUEUE`. Si le fichier n'existe pas, le script crée lui-même les deux variables :

```
if [ -f /etc/sysconfig/sendmail ] ; then
        . /etc/sysconfig/sendmail
else
        DAEMON=yes
        QUEUE=1h
fi
```

Si l'environnement du système d'exploitation n'est pas capable de gérer un réseau, on quitte le script :

```
[ ${NETWORKING} = "no" ] && exit 0
```

Si l'exécutable `sendmail` n'existe pas, il est inutile de continuer :

```
[ -f /usr/sbin/sendmail ] || exit 0
```

Le code de retour du script positionné à zéro est signe de réussite :

```
RETVAL=0
```

Le reste du code comporte une solution de remplacement à plusieurs options. Suivant ce que l'utilisateur va utiliser comme paramètres du script, le code propose le démarrage, l'arrêt, le redémarrage ou l'état du service. Tout autre paramètre affiche un petit commentaire sur l'utilisation du script.

Voici le script de démarrage de Sendmail. La base des aliases est reconstruite grâce au programme `newaliases`. La présence des fichiers `virtusertable`, `access`, `domaintable` et `mailertable` dans le répertoire `/etc/mail` est testée. Si le fichier existe, la base correspondante est créée. La construction des bases se fait *via* l'utilitaire `makemap` en mode à accès calculé (`hash`) :

```
start)
      # Start daemons.
      echo -n "Starting sendmail: "
      /usr/bin/newaliases > /dev/null 2>&1
      for i in virtusertable access domaintable mailertable ; do
          if [ -f /etc/mail/$i ] ; then
              makemap hash /etc/mail/$i < /etc/mail/$i
          fi
      done
      daemon /usr/sbin/sendmail $([ "$DAEMON" = yes ] && echo -bd) \
                                $([ -n "$QUEUE" ] && echo -q$QUEUE)
      RETVAL=$?
      echo
      [ $RETVAL -eq 0 ] && touch /var/lock/subsys/sendmail
      ;;
```

Le programme Sendmail est ensuite lancé *via* la fonction `daemon`, avec les options indiquées au démarrage du script. En cas de réussite, le fichier clé `/var/lock/subsys/sendmail` est crée :

```
stop)
      # Stop daemons.
      echo -n "Shutting down sendmail: "
      killproc sendmail
      RETVAL=$?
      echo
      [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/sendmail
      ;;
```

L'arrêt de Sendmail s'effectue *via* la fonction `killproc`. L'utilisation de cette fonction pour Sendmail est toutefois discutable. En effet, lorsque vous examinez son code, vous constatez qu'en l'absence de paramètre, la fonction peut détruire le processus au moyen

d'un `kill -9`, ce qui n'est pas sans danger pour les messages qui sont en train d'être gérés par Sendmail.

Si l'exécution de la fonction `killproc` est un succès, effacez le fichier clé `/var/lock/subsys/sendmail` :

```
restart|reload)
      $0 stop
      $0 start
      RETVAL=$?
      ;;
```

Pour le redémarrage de Sendmail, prenez le paramètre `$0`, qui correspond à `sendmail`, et appelez les fonctions `stop()` puis `start()`. La variable `RETVAL` contient le code de retour de la fonction `start()` :

```
status)
      status sendmail
      RETVAL=$?
      ;;
```

Ces lignes testent la présence de Sendmail. La fonction `status` teste la présence des deux fichiers suivants :

- `/var/run/sendmail.pid`, qui contient l'identificateur de Sendmail ainsi que l'instruction exacte qui a permis le lancement du service.
- `/var/lock/subsys/sendmail`, qui indique que Sendmail a été lancé ultérieurement au moyen d'une instruction `service sendmail start`.

La variable `RETVAL` contient le code de retour du script `status`.

En utilisation par défaut, les lignes suivantes affichent la manière d'utiliser le script. Pour en sortir, positionnez le code de retour `SHELL` à `1` :

```
*)
      echo "Usage: sendmail {start|stop|restart|status}"
      exit 1
```

Voici la sortie, avec le code de retour obtenu suivant la commande utilisée :

```
exit $RETVAL
```

On peut regretter que le script ne comporte pas d'option de relecture de la configuration.

Sendmail en ligne de commande

Outre sa principale fonctionnalité d'agent de transport du courrier, ou MTA (Mail Transfer Agent), Sendmail peut être utilisé pour les tâches suivantes :

- **Logiciel SMTP client.** Il peut ainsi envoyer du courrier, à l'instar de n'importe quel programme gérant l'envoi des messages électroniques. Il faut toutefois reconnaître que là n'est pas sa véritable finalité et que sa principale utilisation reste le transport.

L'envoi de courrier se fait de la façon suivante :

```
$ /usr/sbin/sendmail moi@labas.com
```

Suit le message lui-même. La fin de ce dernier est précisée grâce à l'indicateur de fin de message, le point unique (.).

Vous pouvez également utiliser la redirection des entrées-sorties :

```
$ /usr/sbin/sendmail moi@labas.com <monmessage
```

Prenez garde que les en-têtes SMTP ne sont pas renseignés et que c'est Sendmail lui-même qui les insère *via* les informations contenues dans le fichier de configuration et d'autres données obtenues par le biais de fonctions système.

- **Gestionnaire de la file d'attente des messages.** Cette fonction est normalement dévolue à Sendmail, mais, dans certaines situations, il peut être intéressant de gérer manuellement le contenu de la file d'attente.
- **Outil de mise au point.** Sendmail comporte un très grand nombre d'options pour la mise au point du système de messagerie ainsi que plusieurs possibilités de tests.
- **Autres tâches secondaires.** Ces tâches incluent la création des bases `aliases`, la visualisation du contenu de la file d'attente, etc.

Les paramètres de Sendmail

Comme expliqué précédemment, Sendmail est appelé simplement de la manière suivante :

```
$ /usr/sbin/sendmail
```

Il est possible d'utiliser des paramètres lors du lancement de Sendmail. Ceux-ci, qui commencent toujours par le caractère -, permettent l'affinage de l'exécution du programme ainsi que la basculement vers ses fonctionnalités secondaires.

Le tableau 6.1 récapitule l'ensemble des paramètres de Sendmail.

Tableau 6.1 Paramètres de lancement de Sendmail

Paramètre	Fonction
`-Btype`	Précise le type du corps du message. Cela correspond à la manière d'encoder les messages. Les valeurs possibles sont `7BIT` ou `8BITMIME`.
`-ba`	Exécution en mode Arpanet. Toutes les lignes en entrée doivent se terminer par les caractères `CR-LF`. Tous les messages sont générés avec les caractères `CR-LF` à la fin. Les en-têtes `From:` et `Sender:` sont également examinés afin de déterminer le nom de l'expéditeur.
`-bd`	S'exécute en tant que service. Sendmail est alors lancé en tâche de fond et s'installe sur le port 25 dans l'attente de connexions SMTP entrantes. Ce mode nécessite les IPC de Berkeley.
`-bD`	identique à `-bd` mais n'est pas exécuté en tâche de fond.
`-bh`	Affiche la base de données des machines persistantes.
`-bH`	Efface les entrées périmées de la base de données des machines persistantes.

Tableau 6.1 Paramètres de lancement de Sendmail *(suite)*

Paramètre	Fonction
`-bi`	Initialise la base de données des aliases.
`-bm`	Livre le courrier normalement (par défaut).
`-bp`	Affiche le contenu de la file d'attente.
`-bs`	Utilise sur les entrées et sorties standards le protocole SMTP normalisé par la RFC 821. Cette option implique que toutes les opérations effectuées *via* l'option `-ba` soient compatibles avec le protocole SMTP.
`-bt`	Exécution en mode de test d'adresses. Ce mode lit les adresses et montre les différentes étapes de l'analyse. Cela se révèle utile pour mettre au point les tables de configuration.
`-bv`	Vérifie seulement les noms et ne tente pas d'accepter ou de livrer un message. Le mode de vérification est normalement utilisé pour valider des utilisateurs ou des listes de diffusion.
`-Cfichier`	Utilise un fichier de configuration différent. Sendmail n'est pas exécuté en tant que `root` si le fichier de configuration principal n'est pas utilisé.
`-dX`	Fixe la valeur de mise au point à `X` (voir plus loin).
`-Fnom`	Précise le nom de l'expéditeur.
`-fnom`	Précise le nom de la personne dans le champs `From:`. C'est ce dernier qui est utilisé lors de l'envoi d'accusé de réception et qui peut apparaître dans le champ `Return-Path:`.
`-G`	Soumission de relayage d'un message
`-hN`	Fixe le nombre de tronçons à `N`. Lorsque le message traverse un MTA, le compteur de tronçon est incrémenté. Lorsque celui-ci atteint sa limite, le message est considéré comme perdu. Cela permet de détecter les boucles sans fin.
`-i`	Ignore les points uniques (`.`) dans les messages entrants. Le point unique est le caractère signalant la fin d'un message. Cette option doit être utilisée lorsque les données à lire proviennent d'un fichier.
`-Létiquette`	Fixe l'étiquette de l'identificateur à utiliser lors de l'envoi des messages vers le service `syslog`.
`-N dsn`	Fixe à `dsn` les notifications de l'état de livraison. Ces notifications peuvent être `never` pour ne pas générer de notification ou une liste séparée par des virgules contenant les valeurs suivantes : – `failure` lors de l'échec de la livraison ; – `delay` lors d'un retard de livraison ; – `success` lors de la réussite de la livraison.
`-n`	Ne gère pas l'aliasing.
`-O option=valeur`	Fixe l'option à la valeur indiquée. Le format utilisé correspond à un mot-clé précis. Voir l'annexe B, pour plus de détails sur les options M4.
`-ox valeur`	Fixe l'option `x` à la valeur indiquée. Le format utilisé correspond à un seul caractère.
`-pprotocole`	Indique le nom du protocole à utiliser pour recevoir le message. Cela peut être un nom simple de protocole, comme `UUCP`, par exemple, ou un binôme formé du protocole et du nom de la machine, tel `UUCP:machine`.
`-q[temps]`	Gère les messages sauvegardés dans la file d'attente suivant un intervalle de temps donné. Si le temps n'est pas précisé, la file d'attente n'est gérée qu'une fois. Le temps se présente sous la forme d'une chaîne de caractères composée de un ou plusieurs chiffres ainsi que d'unités. Celles-ci peuvent être un `s` pour les secondes, un `m` pour les minutes, un `h` pour les heures, un `d` pour les jours et un `w` pour les semaines. Par exemple, `-q15m` demande à Sendmail de gérer les messages déposés dans la file d'attente toutes les quinze minutes.
`-qIchaîne`	Limite les travaux à ceux qui contiennent la chaîne de caractères `chaîne` dans l'identificateur de la file d'attente.

Tableau 6.1 Paramètres de lancement de Sendmail *(suite)*

Paramètre	Fonction
`-qRchaîne`	Limite les travaux à ceux contenant la chaîne de caractères `chaîne` dans l'adresse du destinataire, par exemple `-qRmoi.com`.
`-qSchaîne`	Limite les travaux à ceux qui contiennent la chaîne de caractères `chaîne` dans l'adresse de l'émetteur, par exemple `-qStoi.com`.
`-R retour`	Lors d'une erreur de livraison, limite la portion du message à retourner à l'expéditeur. Les paramètres peuvent être `full`, et dans ce cas tout le message est retourné, ou `hdrs`, pour retourner uniquement les en-têtes.
`-rnom`	Une autre forme périmée du paramètre `-fnom`
`-t`	Lit les messages envoyés aux destinataires. Les en-têtes `To:`, `Cc:` et `Bcc:` sont analysés. L'en-tête `Bcc:` est effacé avant expédition du message.
`-U`	Soumission initiale. Cette option doit toujours être indiquée lorsqu'un agent utilisateur tel que `Mail` ou `Exmh` est employé. Par contre, elle ne doit surtout pas être utilisée lorsque Sendmail est appelé par un agent de livraison tel que `Rmail`.
`-V envid`	Fixe l'identificateur initial de l'enveloppe. Celui-ci est propagé aux serveurs supportant les DSN et est retourné dans les message d'erreur `dsn`.
`-v`	Exécution en mode verbeux. Beaucoup plus d'informations sont affichées lors du traitement.
`-X`	Fichier log. Enregistre tout le trafic entrant et sortant dans le fichier indiqué. Cette option est utilisée lors de la recherche de problèmes et dans la mise au point du service.
`--`	Stoppe la gestion des paramètres et considère le reste des arguments comme des adresses.

Voici quelques exemples d'utilisation de ces paramètres.

Pour initialiser la base des aliases en mode verbeux :

```
# /usr/sbin/sendmail -bi -v
```

Pour forcer l'expédition des messages à destination du domaine `tutu.com` en mode verbeux :

```
# /usr/sbin/sendmail -qRtutu.com -v
```

Pour envoyer un message à `alain@titi.com` depuis l'adresse `root@tutu.com` :

```
# /usr/sbin/sendmail -froot@tutu.com alain@titi.com
```

Pour vérifier que vous pouvez envoyer un message à `root@titi.com` :

```
# /usr/sbin/sendmail -bv root@titi.com
```

Pour lancer le service sans rendre la main à l'utilisateur, la file d'attente étant gérée toutes les heures :

```
# /usr/sbin/sendmail -bD -q1h
```

Les mises au point

Les mises au point sont utilisées dans le cadre de la résolution d'incidents et de recherche de problèmes.

Sendmail dispose d'une quantité impressionnante de tests embarqués, qui permettent de suivre précisément tout type d'opération. Ces mises au point se font par l'intermédiaire du paramètre `-d`, suivi de la catégorie de l'événement, d'un point ainsi que de son niveau.

En voici un exemple :

```
# /usr/sbin/sendmail -d0.1 -v tutu@tutu.com <dev/null
```

```
Version 8.11.4
 Compiled with: LOG MATCHGECOS MIME7TO8 MIME8TO7 NAMED_BIND NETINET
                NETUNIX NEWDB QUEUE SCANF SMTP USERDB XDEBUG

============ SYSTEM IDENTITY (after readcf) ============
      (short domain name) $w = localhost
  (canonical domain name) $j = s1.d1.fr
         (subdomain name) $m = localdomain
              (node name) $k = s1
========================================================
```

Les sorties obtenues *via* cette option donnent beaucoup de renseignements utiles pour la résolution des problèmes. Ici, le paramètre renseigne sur le numéro de version de Sendmail (8.11.4).

La catégorie de l'événement est un numéro privé correspondant aux dispositifs internes de Sendmail : `0` pour les fonctions principales, `17` pour les livraisons, `27` pour les aliases, etc.

Parmi la centaine de tests disponibles, seuls quelques-uns sont vraiment utiles. Le tableau 6.2 recense ces derniers.

Tableau 6.2 Tests de mise au point

Paramètre	Fonction
`-d0.1`	Affiche la version.
`-d0.4`	Nom et alias
`-d0.15`	Trace les agents de livraison.
`-d0.20`	Affiche l'adresse réseau de chaque interface.
`-d4.80`	Trace la fonction `enoughspace()`.
`-d6.1`	Affiche les messages en échec.
`-d8.1`	Échec de la recherche du MX
`-d8.2`	Appel de la fonction `getcanonname()`
`-d8.3`	Trace les noms d'hôtes locaux en échec.

Tableau 6.2 Tests de mise au point *(suite)*

Paramètre	Fonction
`-d8.5`	Les noms d'hôtes essayés par la fonction `getcanonname()`
`-d8.7`	Réponse Oui/Non à `-d8.5`
`-d8.8`	Résolution MX en échec
`-d11.1`	Trace la livraison.
`-d11.2`	Montre le UID/GUID durant la livraison.
`-d12.1`	Trace une machine.
`-d13.1`	Montre la livraison.
`-d20.1`	Montre la résolution de l'agent de livraison.
`-d21.2`	Trace les macros `$&`.
`-d22.1`	Trace l'instanciation des adresses.
`-d22.11`	Montre les adresses avant analyse.
`-d25.1`	Trace `sendtolist`.
`-d27.1`	Trace l'aliasing.
`-d27.2`	Inclut les fichiers et les erreurs sur le répertoire `home`.
`-d27.3`	Chemin de transmission et attente de l'alias
`-d27.4`	Affichage non sécurisé
`-d27.9`	Montre les changements de GUID/UID avec les lectures des `:include`.
`-d28.1`	Affiche les transactions des bases de données.
`-d29.4`	Affiche les correspondances embrouillées.
`-d31.2`	Trace la gestion des en-têtes.
`-d34.11`	Trace la génération et l'omission des en-têtes.
`-d35.9`	Affiche les valeurs des macros prédéfinies.
`-d37.1`	Trace les options.
`-d37.8`	Affiche l'ajout de mots à une classe.
`-d38.2`	Montre l'ouverture des bases ainsi que leur échec.
`-d38.4`	Montre le résultat d'une base ouverte.
`-d38.19`	Trace le basculement vers une base.
`-d38.20`	Trace la résolution des bases.
`-d41.1`	Trace les commandes de la file d'attente.
`-d44.5`	Trace la fonction `writable()`.
`-d48.2`	Trace les appels vers les groupes de vérification des règles.
`-d60.1`	Trace les résolutions des bases à l'intérieur de la fonction `rewrite()`.
`-d99.100`	Empêche que le service soit exécuté en tâche de fond.

Voici un exemple de vérification de l'aliasing :

```
# /usr/sbin/sendmail -d27.1 root@tutu.fr </dev/null
alias(root)
root@tutu.fr (, root) aliased to root@mail.tutu.fr
self_reference(root@mail.tutu.fr)
forward(root@tutu.fr)
```

La file d'attente

Lorsqu'un message est géré par Sendmail, il existe deux situations possibles :

- Le message peut être livré immédiatement, et Sendmail l'expédie sans préavis.
- Le message ne peut être livré pour différentes raisons :
  - Le serveur de destination n'est pas accessible.
  - Sendmail ne dispose pas de suffisamment de ressources pour mener à bien l'expédition.
  - Sendmail est paramétré pour déférer systématiquement le transport du courrier, un autre processus se chargeant du transport ultérieurement.

Dans le second cas, les messages sont déposés dans une file d'attente gérée périodiquement par Sendmail.

Emplacement de la file d'attente

La file d'attente n'est rien d'autre qu'un répertoire contenant un ensemble de fichiers formant le message électronique. L'emplacement de ce répertoire est défini grâce à l'option `QueueDirectory`, comme ci-dessous :

```
O QueueDirectory=/var/spool/mqueue
```

ou, en M4 :

```
define(`QUEUE_DIR',`/var/spool/mqueue')
```

Notez que c'est généralement le répertoire `/var/spool/mqueue` qui est utilisé. Il est aussi possible d'utiliser un répertoire à sa convenance, comme ici :

```
define(`QUEUE_DIR',`/var/spool/file')
```

Gestion de la file d'attente

La file d'attente est gérée périodiquement par Sendmail, qui tente à intervalle régulier d'expédier tout le contenu de la file vers les destinataires. L'intervalle de gestion de cette file d'attente est défini au démarrage de Sendmail *via* la commande `-q`.

Cette dernière demande un paramètre, lequel détermine l'intervalle d'exécution. Si vous omettez ce paramètre, la file d'attente est gérée immédiatement.

La commande ci-dessous déclenche la gestion de la file d'attente :

```
# /usr/sbin/sendmail -q
```

Les options `Timeout.queuewarn` et `Timeout.queuereturn` contiennent la durée maximale pendant laquelle Sendmail conserve le message sans générer d'exception.

Une première exception est créée lorsque la durée de rétention du message dépasse le délai défini par `Timeout.queuewarn`. Un message d'avertissement est alors expédié à l'émetteur. Dans un second temps, au-delà du délai fixé par `Timeout.queuereturn`, un message d'erreur est expédié à l'émetteur, et le message est détruit et supprimé de la file d'attente par Sendmail.

Pour l'exécution régulière de la gestion de la file d'attente, il est préférable de déterminer un laps de temps qui s'accommode de la charge et du degré de disponibilité du serveur. Un délai situé entre un quart d'heure et une heure est raisonnable pour la plupart des situations.

Le délai est spécifié de la façon suivante :

```
#  /usr/sbin/sendmail -q15m
```

Cette commande indique que la file d'attente est examinée et gérée par Sendmail toutes les quinze minutes. Le délai d'exécution est récupérable à partir de la macro `${queue_interval}`. La macro `${daemon_info}` peut également donner cette information.

Lors du démarrage de Sendmail, vous devez lui fournir deux indications vitales :

- Qu'il accepte le courrier en entrée, ce qui revient à lui faire écouter le trafic sur le port SMTP (25).
- Qu'il gère périodiquement les messages bloqués pour les raisons évoquées précédemment.

En ligne de commande, cela se traduit de la façon suivante :

```
# /usr/sbin/sendmail -bd -q15m
```

L'option `-bd` demande à Sendmail de s'exécuter comme service SMTP, et l'option `-q15m` précise l'intervalle de gestion de la file d'attente.

Autre gestion de la file d'attente

En plus de l'exécution régulière de la gestion de la file d'attente, il peut être utile de lancer une autre instance de Sendmail à une heure de moindre influence et de tenter de vider la file d'attente grâce à des paramètres d'expédition plus agressifs.

Vous pouvez, par exemple, réduire les délais impartis aux différents événements de la transaction SMTP pour éviter que Sendmail ne bloque sur la livraison difficile d'un message. Il est possible de limiter le temps d'attente d'une commande SMTP, telle que `RCPT TO`, `MAIL TO`, etc., et de se défaire de la sorte plus rapidement de la gestion d'un message au profit de ceux qui présentent moins de difficulté à être livrés.

Dans ce cas, les paramètres intéressants sont les suivants :

```
Timeout.initial Timeout.connect Timeout.iconnect
Timeout.helo  Timeout.mail  Timeout.rcpt
Timeout.datainit Timeout.datablock Timeout.datafinal
Timeout.rset  Timeout.quit  Timeout.misc
Timeout.command Timeout.ident
```

La signification de ces paramètres est donnée au tableau 6.3.

Tableau 6.3 Paramètres de gestion de la file d'attente

Paramètre	Signification
`Timeout.initial`	Donne le délai d'attente d'une réponse sur une connexion initiale.
`Timeout.connect`	Délai d'attente avant qu'un `connect()` initial ne s'achève. Cela permet de raccourcir les délais de connexion. Le noyau UNIX impose un maximum, qui peut varier d'un système à un autre.
`Timeout.iconnect`	Agit comme `Timeout.connect` mais uniquement sur la première tentative de connexion vers un serveur. Cela permet d'effectuer une première passe rapide suivie de tentatives de livraison plus raisonnables.
`Timeout.helo`	Délai d'attente pour obtenir une réponse à la commande `HELO` ou `EHLO`
`Timeout.mail`	Délai d'attente d'une réponse à la commande `MAIL`
`Timeout.rcpt`	Délai d'attente d'une réponse à la commande `RCPT`
`Timeout.datainit`	Délai d'attente d'une réponse `354` à la commande `DATA`
`Timeout.datablock`	Délai d'attente d'un bloc d'information durant la phase `DATA`
`Timeout.datafinal`	Délai d'attente d'une réponse au point final `.`, qui détermine la fin d'un message.
`Timeout.rset`	Délai d'attente d'une réponse à la commande `RSET`
`Timeout.quit`	Délai d'attente d'une réponse à la commande `QUIT`
`Timeout.misc`	Délai d'attente d'une réponse à toute autre commande
`Timeout.command`	Délai d'attente après l'exécution de chaque commande
`Timeout.ident`	Délai d'attente d'une réponse après une requête `IDENT` (RFC 1413)

La gestion de la file d'attente dotée des nouveaux paramètres peut être lancée grâce au service d'exécution par lot (`crond`), ici tous les jours à 23 h 55 :

```
55 23 * * * /usr/sbin/sendmail -q -C/etc/mail/sendmail-soir.cf >/dev/null </dev/null
```

Le fichier `/etc/mail/sendmail-soir.cf` est une configuration adaptée à ces besoins et dotée de paramètres « agressifs ». Il indique à Sendmail qu'il doit gérer plus vite le transport des messages.

Voici le fichier `sendmail.mc` contenant cette configuration :

```
include(`/usr/lib/sendmail-cf/m4/cf.m4')
VERSIONID(`Sendmail pour relancer la queue SMTP')dnl
OSTYPE(`linux')dnl
define(`confMAX_MESSAGE_SIZE', `15000000')dnl
define(`STATUS_FILE', `/etc/mail/sendmail.st')dnl
```

```
define(`confDOMAIN_NAME', `mail1.dO.com')dnl
define(`ALIAS_FILE', `/etc/mail/aliases')dnl
define(`confTO_INITIAL',`1m')dnl
define(`confTO_CONNECT',`1m')dnl
define(`confTO_DATABLOCK', `1m')dnl
define(`confTO_DATAFINAL', `2m')dnl
define(`confTO_COMMAND', `1m')dnl
define(`confTO_HELO', `1m')dnl
define(`confTO_MAIL', `1m')dnl
define(`confTO_RCPT', `1m')dnl
define(`confTO_DATAINIT', `1m')dnl
define(`confTO_RSET', `1m')dnl
define(`confTO_QUIT', `1m')dnl
FEATURE(`access_db',`hash -o /etc/mail/access')dnl
FEATURE(`mailertable',`hash -o /etc/mail/mailertable')dnl
FEATURE(always_add_domain)dnl
FEATURE(local_procmail)dnl
MAILER(smtp)dnl
```

Sous la version 8.11.6 de Sendmail, les paramètres par défaut des délais sont les suivants :

- `Timeout.initial` : 5 min ;
- `Timeout.connect` : 5 min ;
- `Timeout.helo` : 5 min ;
- `Timeout.mail` : 10 min ;
- `Timeout.rcpt` : 1 h ;
- `Timeout.datainit` : 5 min ;
- `Timeout.datablock` : 1 h ;
- `Timeout.datafinal` : 1 h ;
- `Timeout.rset` : 5 min ;
- `Timeout.quit` : 2 min ;
- `Timeout.command` : 1 h.

Grâce à la réduction des principaux délais système, la configuration de Sendmail proposée ci-dessus garantit une nette accélération du transport des messages. Même sur une connexion difficile, une transaction n'excède pas quelques minutes.

Gestion avancée de la file d'attente

Il est parfois intéressant d'utiliser l'option `-v` lors de la gestion de la file d'attente. Celle-ci affiche le dialogue SMTP sur la sortie courante, ce qui permet de visualiser les éventuelles erreurs susceptibles de se produire :

```
# /usr/sbin/sendmail -q -v
Running MAA010302 (sequence 1 of 10)
```

```
<moi@toi.com> ... Connecting to mail.toi.com
Trying 192.168.10.1 Connection time out
<moi@toi.com>... Deferred: mail.toi.com timed out.
```

Vous pouvez de la sorte déterminer aisément les connexions ou livraisons « à problèmes ». Grâce à d'autres options élaborées de gestion de la file d'attente, Sendmail permet de privilégier la livraison vers certains destinataires ou en provenance de certains émetteurs. Cela se fait au moyen des options `-qRdestinataire` et `-qSemetteur`.

La commande suivante indique que Sendmail doit gérer les messages à destination du domaine `moi.com` :

```
#  /usr/sbin/sendmail -qRmoi.com -v
```

Cela se révèle particulièrement utile lorsque vous souhaitez forcer l'expédition du courrier vers ou depuis un domaine privilégié. La même gestion fine de la file d'attente peut être effectuée en se fondant sur l'identificateur du message.

Cela s'effectue par le biais de l'option `-qIidentificateur`. La commande suivante déclenche la gestion des messages contenant l'identificateur `AAA` :

```
#  /usr/sbin/sendmail -qIAAA -v
```

Les éléments de la file d'attente

Si vous examinez le contenu de la file d'attente, vous trouvez une succession de fichiers, comme ci-dessous :

```
tff4TDCqhO2575
dff4TCnK301830
dff4TCeC301684
tff4TC4m605015
xff4TC4L605009
dff4TC3j605002
dff4TC4m605015
dff4TC4L605009
xff4TC3j605002
xff4TC1f604946
tff4TB7kDO3740
tff4RKHMI31676
xff4RKHMI31676
tff4RIidI25536
Tff4QKnXIO5439
xff4QKo7IO5553
tff4QKnZIO5486
xff4QKnZIO5486
dff4QKniIO5504
xff4QKnjIO5506
xff4QKnnIO5525
dff4QKncIO5492
Tff4QIdIIO4238
```

Ces fichiers forment l'ensemble des messages électroniques en instance d'expédition. Sendmail les gère en séparant les en-têtes du corps et en créant des fichiers temporaires et annexes pour ses propres besoins internes.

Les fichiers antérieurs à la date spécifiée par l'option `Timeout.queuereturn` (cinq jours par défaut) peuvent être considérés comme perdus. Ils proviennent à coup sûr d'un arrêt intempestif de Sendmail, qui était en train de les gérer et qui n'a pas eu le temps de clôturer correctement l'opération. Généralement, il s'agit d'un arrêt brutal du service SMTP (`kill -9` sur Sendmail) ou d'une extinction brutale de la machine.

Un fichier présent dans la file d'attente est toujours unique et se construit de la manière suivante :

```
Xfidentificateur
```

`X` représente la nature du fichier (corps, en-tête, verrou), `f` est une constante invariable, et `identificateur` est l'étiquette SMTP unique attachée à chaque message.

La nature du fichier peut être l'un des éléments ci-dessous :

- `.df` pour le corps du message ;
- `.lf` pour le fichier de verrouillage (obsolète) ;
- `.nf` pour le fichier de création de l'identificateur de message (obsolète) ;
- `.tf /Tf` pour les fichiers temporaires ;
- `.xf` pour les fichiers de transcription ;
- `.qf /Qf` pour le fichier de contrôle de la file d'attente.

Les fichiers de verrouillage empêchent que plusieurs instances de Sendmail ne gèrent simultanément le même message.

Les fichiers de type `qf` contiennent les en-têtes du message ainsi que des lignes de contrôle. Ces dernières sont récapitulées au tableau 6.4.

Tableau 6.4 Lignes de contrôle des fichiers `qf`

Code	Description
`B`	Type du corps du message (`8BITMIME` ou `7BIT`)
`C`	Utilisateur de contrôle
`D`	Fichier contenant les données
`E`	Destinataire des notifications d'erreur
`F`	Bits de drapeaux
`H`	Définition des en-têtes
`I`	Numéro de `inode` des fichiers `df`
`K`	Heure de la dernière gestion du message
`M`	Explique pourquoi le message a été placé dans la file.
`N`	Nombre de tentatives

Tableau 6.4 Lignes de contrôle des fichiers `qf` *(suite)*

Code	Description
`P`	Priorité actuelle
`Q`	Destinataire original
`R`	Adresse du destinataire
`S`	Adresse de l'émetteur
`T`	Heure de création
`V`	Version
`Z`	Identificateur de l'enveloppe DSN
`$`	Restaure la valeur de la macro.
`.`	Fin du fichier `qf`

L'identificateur se compose de l'heure au format numérique, suivie du numéro de processus. Cet identificateur se retrouve également à l'intérieur de la macro `$i`.

Le contenu exact de la file d'attente est visualisable grâce à la commande `mailq` ou au paramètre `–bp (/usr/sbin/sendmail -bp)`, comme ici :

```
----Q-ID---- --Size-- -----Q-Time----- ------------Sender/Recipient------------
g33DxEV31626      536 Wed Apr  3 15:59 <nobody@d0.com>
                 (Deferred: Connection refused by d1.com.)
                                       <database@d1.com>
g33DWLV27345*   35176 Wed Apr  3 15:32 <callcenter@d3.com>
                 (host map: lookup (d4.fr):)
                                       <contact@d4.fr>

g33DNjV25499      469 Wed Apr  3 15:23 <nobody@d5.be>
                 (Deferred: Connection refused by d6.com.)
                                       <database@d6.com>
```

On retrouve, dans l'ordre, l'identificateur du message, la taille, l'heure à laquelle le message a été déposé dans la file d'attente, les émetteurs et destinataires, ainsi que quelques commentaires sur l'état d'expédition du message.

L'identificateur peut être suivi d'un caractère indiquant l'état du message :

- Un astérisque (*) signifie que le message est en train d'être géré par Sendmail
- Un signe moins (-) indique que le message ne peut être géré immédiatement car il est trop récent.

Les files d'attente multiples

Depuis la version 8.10, Sendmail dispose, avec les files d'attente multiples, d'un dispositif très intéressant.

L'administrateur spécifie une série de répertoires, qui seront utilisés par le système de gestion de la file d'attente de Sendmail. Le répertoire utilisé par l'instance de Sendmail est choisi de façon aléatoire.

Ce dispositif permet une meilleure répartition de la charge de Sendmail lorsque celui-ci gère sa file d'attente et garantit l'optimisation de l'espace de stockage ainsi qu'une accélération de l'accès au contenu des répertoires.

L'utilisation de files d'attente multiples se fait très simplement par le biais de l'instruction M4 suivante :

```
define(QUEUE_DIR,`/var/spool/mqueue/f*')
```

Sendmail utilise chaque répertoire correspondant au paramètre de l'option `QUEUE_DIR`. Dans l'exemple ci-dessus, tout répertoire commençant par la lettre `f` dans `/var/spool/mqueue` est utilisé.

Il va de soi que vous devez créer manuellement ces répertoires, accompagnés des bons droits d'accès :

```
# cd /var/spool/mqueue
# mkdir file1
# mkdir file2
# mkdir file3
# chmod 700 *
```

Ici, les différentes files d'attente sont réservées à l'usage exclusif du superutilisateur `root`. Lors du redémarrage, Sendmail utilise toutes ces files d'attente.

Les avantages de ce dispositif sont les suivants :

- Les files d'attente utilisées peuvent être réparties sur plusieurs espaces de stockage, ce qui permet d'envisager un meilleur équilibrage de charge.
- Lors de l'exécution programmée du gestionnaire de file d'attente de Sendmail, une instance du programme est utilisée pour chacune des files d'attente. Dans notre exemple, trois instances de Sendmail s'occupent de gérer les messages. Cela a pour effet de diviser par trois le temps d'exécution de la tâche.
- Bon nombre de systèmes de fichiers non journalisés fonctionnent de façon optimale avec des répertoires contenant plusieurs milliers d'occurrences. Au-delà, le système de parcours des répertoires, principalement la fonction UNIX intégrée `readdir()`, peut mettre un certain temps pour être exécuté. Encore une fois, la multiplicité des files d'attente réduit considérablement le parcours des répertoires et diminue en conséquence le temps d'accès aux éléments contenus dans ces files d'attente. Ce gain de temps est notamment intéressant pour les fournisseurs d'accès Internet, qui doivent souvent gérer plusieurs milliers de messages déposés dans la file d'attente.

Ce système présente toutefois le défaut de consommer beaucoup de mémoire vive. Lors de l'exécution de la fonction de gestion de la file d'attente, un nombre plus important d'instances de Sendmail est créé. Si, de surcroît, l'administrateur opte pour l'option `confSEPARATE_PROC`, qui crée un processus par instance de livraison, le système peut se trouver en difficulté, faute de mémoire. Ce système n'est donc pas recommandé sur les machines dotées de peu de mémoire vive.

En résumé

Il existe deux façons de démarrer et d'arrêter Sendmail : *via* des outils conventionnels UNIX ou au moyen de scripts développés par Redhat.

Sendmail dispose de très nombreux paramètres, qui permettent d'altérer son exécution et de lancer toute une batterie de tests. La file d'attente est gérée régulièrement par Sendmail. Elle contient l'ensemble des messages en instance de livraison.