CONFIGURAZIONE DEL SERVER DI POSTA ELETTRONICA DELL'OSSERVATORIO ASTRONOMICO DI TORINO

Rapporto Tecnico n. 084
Date: 7 Dicembre 2006
Autori: A.Cora, F.Salvati

INDICE

INTRODUZIONE

Con la ristrutturazione dei servizi di calcolo dell'Osservatorio Astronomico di Torino, si e` colta l'occasione per redigere un manuale dettagliato di installazione del Servizio di Posta Elettronica rivolto a chi si propone di cofigurare un server di posta elettronica e al system manager che deve mantenerne l'operatività.
In questo documento illustriamo l'installazione e configurazione dei vari software, nell'ultima parte affrontiamo i possibili problemi che possono intervenire e gli aspetti della manutenzione software del servizio.
Le istruzioni qui contenute, pur facendo riferimento alla piattaforma UNIX-like da noi utilizzata (LINUX FEDORA CORE4) ed alla nostra configurazione locale, hanno validita` generale per qualsiasi piattaforma LINUX.

CONVENZIONE

Nella redazione del documento, si è utilzzata le seguente convenzione tipografica, tutto ciò che appare a schermo è stampato in verde chiaro (comandi e standard output) il resto del testo in caratteri neri.

1.0 ARCHITETTURA DEL SERVIZIO

Il Servizio di Posta Elettronica è sicuramente quello che negli ultimi anni ha assunto maggiore rilevanza per lo svolgimento quotidiano del lavoro di Ricerca e/o Amministrazione.
Ed è paradossale che un operazione semplice qual'è l'invio di un e-mail sia così complicata, infatti l'invio di posta elettronica può essere schematizzato nella scrittura di un file (lettera elettronica) dal computer utilizzato dal mittente in una zona di memoria del computer (casella di posta) accessibile al destinario.
Questa operazione, apparentemente semplice, utilizza vari software di gestione del flusso delle e-mail, e funzionlità sempre più sofisticate che tutelano la sicurezza del computer e la "privacy" dell'utente.
Inoltre le dilaganti piaghe dei Virus e dello Spam, hanno costretto gli amministratori dei sistemi all'adozione di strati di software in grado di garantre appropriate contromisure.
L'architettura di un generico servizio di posta elettronica é riportata in figura n.1.

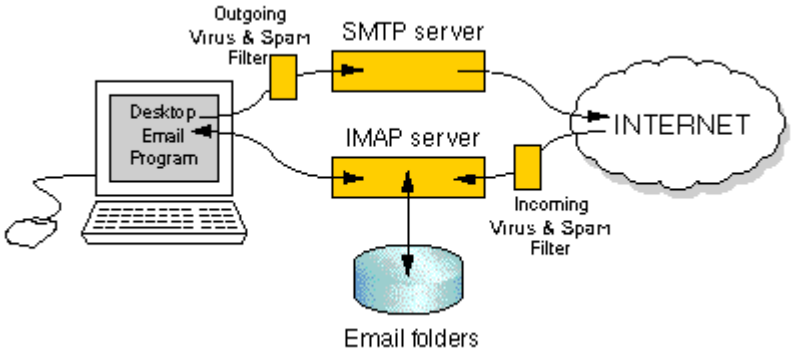

Figura 1- Architettura del servizio di posta elettronica

Sono presenti semplicemente il software in grado di inviare la posta (Outgoing) e un dispositivo in grado di smistare i mail ricevuti (Incoming) . Nei server di posa piuù semplici queste funzionlità son garantite da un unico software quale SENDMAIL.
Descriviamo, ora, l'architettura del servizio del sistema implementato in Osservatorio, schematizzato in figura n.2.
Il cuore del servizio è costituito dal Mail Transfert Agent (MTA), che ha il compito di trasportare la posta elettronica

dal mttente alla destinatario. Tra i possibili MTA si è optato per SENDMAIL che ha la possibilità di implementare il Mail-fILTER (MILTER) necessari al software antivirus e antispam.
SENDMAIL, alloca la posta in casselle poste nella dirctory /var/spool/mail del server, ove l'utente puo' leggerla tramite il login sul proprio account e con le basilari comandi UNIX (mail,mailx) e/o programmi disponibili sul server quali PINE.
SENDMAIL si occupa di inoltrare e ricevere la posta elettronica, non si occupa di rendere accessibile la casella di posta a client differenti ed altri computer in rete, compito assolto dal Local Delivery Agent (LDA) più evoluto in grado di supportare protocolli sicuri.
Abbiamo utilizzato quale LDA: DOVECOT distribuito con LINUX FEDORA CORE4. DOVECOT consente la distribuzione dei mail con protocolli POP (Post Office Protocol), POPS(.....), IMAP(.....) e IMAPS(.....).
L'utente può scegliere tra differenti Mail User Agent (MUA) per la lettura della posta. All'interno dell'Osservatorio gli utenti utilizzano differenti MUA: Netscape, Outlook, Thunderbird Eudora, e Dtmail, questi programmi configurando l'outgoing mail server utilizzeranno l'MTA anche per l'invio della posta.
Inoltre è garantito un accesso alla casella di posta tramite il WEB detto WEBMAIL; questo servizio è basato su Mailman, e la descrizione della sua installazione è esclusa dal presente documento .

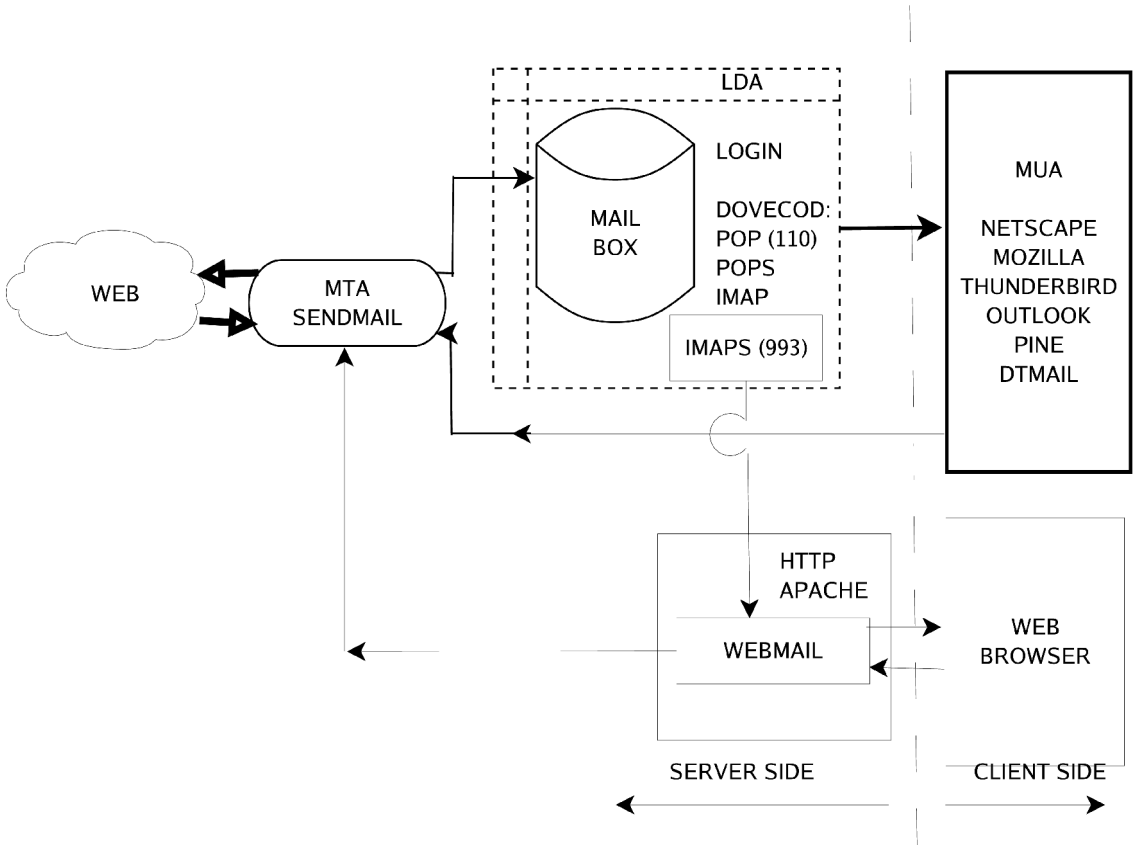

Figura 2- Architettura del servizio di posta elettronica OATo-INAF

Nella realizzazione del server si è tenuto conto di:

- Piena compatibilità con le vecchie mail box e account utenti
- Utilizzo di software Open Source
- Filtro antivirus a livello di server, abilitato al confinamento del mail infetto in un area di quarantena, al fine di evitare di infettare i client
- Filtro antispam a livello di server, in grado di identificare i probabili Spam e marchiarli, lasciando la

possibilità all'utenza di verificare la correttezza dell'identificazione.

- Possibile implementazione di domini virtuali al fine di estendere il servizio ad altri Istituti

In particolare, il primo punto, ovvero mantenere la piena compatibilità con il servizio precedente ha implicato un notevole allungamento dei tempi: l'installazione del server inizia in Aprile 2006, il completamento della migrazione degli utenti si concluderà nel Marzo 2007.

2.0 IL MAIL TRANSFERT AGENT (SENDMAIL)

Sendmail è un software BSD (Berkeley System Dystribution, incluso nella distribuzione di Linux Fedora core 4, non ha quindi bisogno di una vera e propria installazione, se non fosse per la mancanza dei pacchetti quali le librerie MILTER e VACATION.

2.1 SENDMAIL INSTALLAZIONE DELLE LIBRERIE

Il software SENDMAIL è stato installato con LINUX FEDORA CORE4, ma è incompleto, seppure copilato per supportare i MILTER, siamo stati costretti ad installare le librerie (LIBMILTER).
Il test effettuato su sendmail dimostra che è compilato per supportare i MILTER:

```
sendmail -d0.1 -bv

Version 8.13.1
 Compiled with: DNSMAP HESIOD HES_GETMAILHOST LDAPMAP LOG MAP_REGEX
                MATCHGECOS MILTER MIME7TO8 MIME8TO7 NAMED_BIND NETINET NETINET6
                NETUNIX NEWDB NIS PIPELINING SASLv2 SCANF STARTTLS TCPWRAPPERS
                USERDB USE_LDAP_INIT
```

durante l'installazione dell'antivirus CLAMAV (descritta nel pagrafo 3.0 e seguenti) ci siamo accorti che mancano le librerie (LIBMILTER). Scarichiamo sendmail-8.13.6 dal sito de progetto (http://). Scompatto l'archivio compresso e mi sposto nell'appropriata directory e provvedo alla loro installazione:

```
cd /home/alberto/SENDMAIL/sendmail-8.13.6/obj.Linux.2.6.11-1.1369_FC4smp.i686/libmilter

make install   (non necessità make!)

.............
if [ ! -d /usr/include/libmilter ]; then mkdir -p /usr/include/libmilter; else :; fi
install -c -o root -g bin -m 0444 ../../include/libmilter/mfapi.h /usr/include/libmilter/mfapi.h
install -c -o root -g bin -m 0444 ../../include/libmilter/mfdef.h /usr/include/libmilter/mfdef.h
    install -c -o root -g bin -m 0444 libmilter.a /usr/lib
```

e le librerie sono installate.

2.2 CONFIGURAZIONE DI SENDMAIL

La configurazione di sendmail è stata senza dubbio quella più complicata e ha subito varie interazioni, i file di configurazione finali fianali sono riportati per intero nelle appendici, qui di seguito descriveremo l'inserimento delle righe di configurazione discutendo il risultato/obiettivo che si desiderava ottenere.
Il file principale di configurazione del sendmail è /etc/mail/sendmail.mc , detto file è propedeuticco alla compilazione del file di configurazione vero e proprio /etc/sendmail.cf.
Al file di configurazione principale si affiancano dei file secondari access, virtusertable e genericstable che si trovano sempre nella directory /etc/mail. Questi file, che definiscono la configurazione di base del servizio sono riporati nelle Apppendici.

2.3 CONFIGURAZIONE DEI MECCANISMI DI AUTENTICAZIONE

In questo paragrafo descriviamo come si verificano la presenza di meccanismi di autenticazione TLS/SSL e si modifica il file di configurazione principale di sendmail, per istruirlo al suo utilizzo e quali sono i demoni che ci garantiscono 'autenticazione e lo startup. Con il comando già utilizzato nel verificare la corretta compilazione di SENDMAIL verifichiamo se vi sono adeguati meccanismi di autenticazione:

```
sendmail -d0.1 -bv
```

ottengo:

```
Version 8.13.1
 Compiled with: DNSMAP HESIOD HES_GETMAILHOST LDAPMAP LOG MAP_REGEX
                MATCHGECOS MILTER MIME7TO8 MIME8TO7 NAMED_BIND NETINET NETINET6
                NETUNIX NEWDB NIS PIPELINING SASLv2 SCANF STARTTLS TCPWRAPPERS
                USERDB USE_LDAP_INIT

============ SYSTEM IDENTITY (after readcf) ============
     (short domain name) $w = sam
  (canonical domain name) $j = sam.oato.inaf.it
        (subdomain name) $m = oato.inaf.it
            (node name) $k = sam.oato.inaf.it
```

STARTTLS e SASL sono presenti e saranno utilizzati per autenticare il relay e gli utenti. Quindi posso generare i certificati per TSL/SSL, che su Linux Fedora core4 sono nella directory:/etc/pki/tls/certs, a differenza delle altre distribuzione che li pongono in /usr/share/ssl/certs. Per generare i certiicati:

```
make sendmail.pem
```

modifico il file di configurazione di sendmail: sendmail.mc introducendo le seguenti linee finalizzate all'autenticazione e l'utilizzazione dei certificati:

```
dnl #
TRUST_AUTH_MECH(`EXTERNAL DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
```

```
define(`confAUTH_MECHANISMS', `EXTERNAL GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
dnl #
dnl # Rudimentary information on creating certificates for sendmail TLS:
dnl #     cd /usr/share/ssl/certs; make sendmail.pem
dnl # Complete usage:
dnl #     make -C /usr/share/ssl/certs usage
dnl #
define(`confCACERT_PATH',`/etc/pki/tls/certs')
define(`confCACERT',`/etc/pki/tls/certs/ca-bundle.crt')
define(`confSERVER_CERT',`/etc/pki/tls/certs/sendmail.pem')
define(`confSERVER_KEY',`/etc/pki/tls/certs/sendmail.pem')
define(`confCLIENT_CERT',`/etc/pki/tls/certs/sendmail.pem')
define(`confCLIENT_KEY',``/etc/pki/tls/certs/sendmail.pem')
```

ricordandosi dell'opzione per l'ascolto sulla porta smtps:

```
DAEMON_OPTIONS(`Port=smtps, Name=TLSMTA, M=s')dnl
```

ricompilo il tutto

```
make -C /etc/mail
```

e faccio ripartire il server mail

```
service sendmail restart
```
(in alternativa a /etc/init.d/sendmail restart)

definisco che l'SASL authentication daemon parta al boot

```
chkconfig saslauthd on
```

e lo faccio ripartire

```
service saslauthd restart
```

3.0 IL SOFTWARE ANTIVIRUS (CLAMAV)

Alla data odierna i virus non sono una grande minaccia per i sistemi Unix e Unix-like come linux. Sono pochi quelli in circolazione, in gran parte sperimentali sviluppati a scopo di studio. Tuttavia è importante adottare un sistema antivirus per i server di posta che debbono essere anche interrogati da client windows.
CLAMAV è uno tra i prodotti open source più diffusi, è sviluppato in ambiente Unix Unix-like , e per questo offre maggiore garanzie per la protezione del server, possiede caratteristiche interessanti, e viene utilizzato da alcuni siti con elevati volumi di traffico.
CLAMAV non presenta strane dipendenze, oltre alla necessita' di disporre le librerie di milter al fine di integrarlo con sendmail e può essere compilato sulla mmaggior parte delle piattaforme senza problemi.
Per l'installazione sul sistema necessita la generazione di un gruppo e di un utente (clamav), procedura comune

agli altri antivirus su piattaforme Unix-like presi in esame (Amavis e Sophos).
I binari principali sono il demone clamd, clamscan (utilizzabile da riga di comando per la scansione di una directory), clamdscam, clam.sock (che genera il socket per il milter)e freshclan (utilità di aggiornamento del database dei virus).
Clamav è in grado di identificare oltre 20.000 tra virus, worm e troiani ed effettua la ricerca anche su file compressi (archivi zip, rar e tar.gz). Abbiamo adottato CLAMAV, poichè facilmente integrabile in send mail tramite i filtri milter, ma ci risulta che il prodotto sia applicabile anche a Postfix.
Il numero di virus sviluppati e circolanti nel network non accenna a diminuire e il tool fresclam, attivato come job dal crond, può aggiornare il database locale dell'antivirus. Lo schema a blocchi del fuzionamente del pacchetto antivirus tramite i milter è riportato in figura n.3.

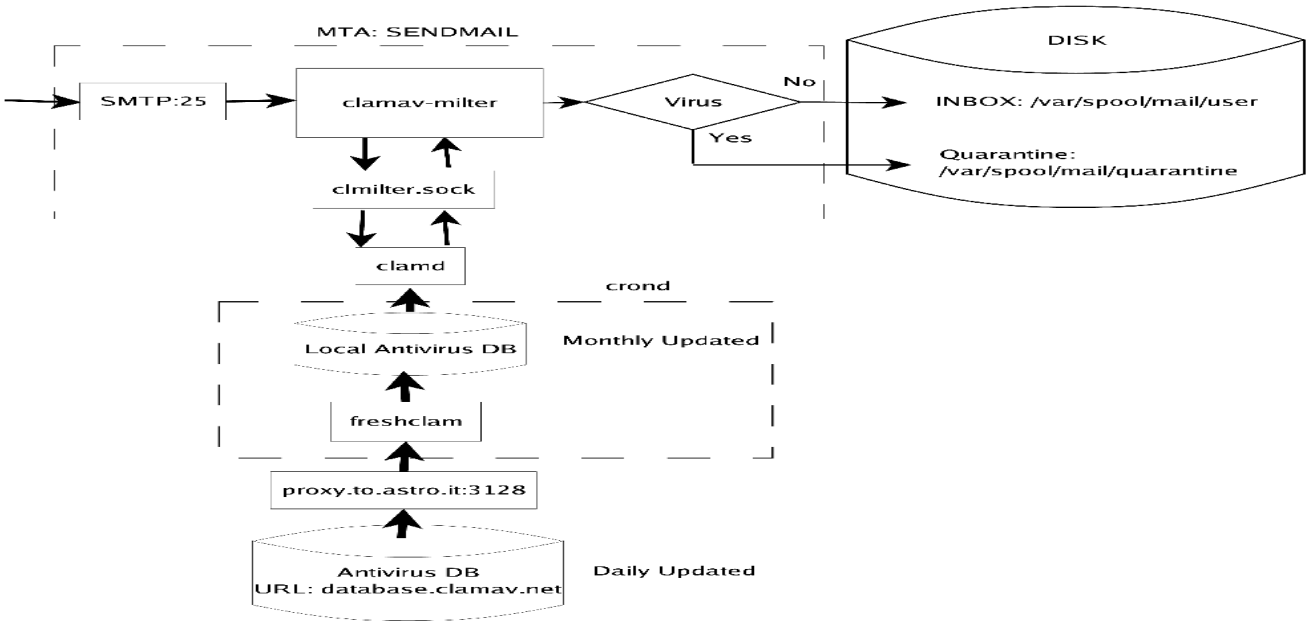

Figura 3: diagarmma a blocchi dell'implementazione dell'antivirus

3.1 INSTALLAZIONE

Scarico il software dal sito http://www.clamav.net la versione : clamav-0.88.1. Con i privilegi di root genero l'utente clamav e il gruppo clamav

```
groupadd clamav
useradd -g clamav -s /bin/false -c "Clam AntiVirus" clamav
```

compilo il software con l'oopzione di abilitazione per i mail-filter (milter)

```
./configure --sysconfdir=/etc --enable-milter
```

nell'ambito della cmpliazione di clamav ci siamo resi conto che mancavano le libreire per i milter, inquesto momento abbiamo scaricato sendamail-8.13.6 e rigenerate le libreie, operazione descritta nel paragrafo 2.1. A video appare

```
checking build system type... i686-pc-linux-gnu
```

```
checking host system type... i686-pc-linux-gnu
checking target system type... i686-pc-linux-gnu
creating target.h - canonical system defines

.............................

config.status: creating docs/man/freshclam.conf.5
config.status: creating clamav-config.h
config.status: executing depfiles commands
```

seguono gli usuali comandi:

```
make
make install
```

installazione riuscita!

3.2 TEST DEL SOFTWARE ANTIVIRUS

L'antivirus è anche operativo a linea di comando: clamscan -option [outputfile] [directory]
Per maggiori opzioni man clamscan ci consente di visualizzare il dettagli dell'utilizzi di clamscan, effettuo una scansione di prova del server digitando:

```
clamscan -r -l scan.txt .
```

Ottengo:

```
---------- SCAN SUMMARY -----------
Known viruses: 48836
Engine version: 0.88.1
Scanned directories: 45
Scanned files: 756
Infected files: 5
Data scanned: 12.93 MB
Time: 5.216 sec (0 m 5 s)
```

3.3 CONFIGURAZIONE CLAMAV MILTER

modifico sendmail.mc introducendo le seguenti linee (per il file sendmail.mc completo si veda l'appendice)

```
INPUT_MAIL_FILTER(`clamav', `S=local:/var/run/clamav/clmilter.sock, F=,T=S:4m;R:4m')dnl
define(`confINPUT_MAIL_FILTERS', `clamav')
```

modifico le proprietà delle directory utilizzate da clamav

```
su
Password:
# mkdir /var/run/clamav/
# chown clamav /var/run/clamav/
# su clamav
# chmod 700  /var/run/clamav/
```

Controllo il file di configurazione di clamav: /etc/clamd.conf che contenga la riga:

```
LocalSocket /var/run/clamav/clamd.sock
```

faccio partire il demone

```
clamd
```

partendo il demone si genera il socket che genera il file: /var/run/clamav/clamd.sock, ma non ancora il milter.sock necessario all'integrazione dell'antivirus su SENDMAIL, per generare questo socket:

```
/usr/local/sbin/clamav-milter -lo /var/run/clamav/clmilter.sock
```

cambio owner in clamav del file di log /tmp/clamd.log

```
chown clamav /tmp/clamd.log
```

per il test sono sufficienti 2 –max-children, ma questo numero di processi dovrà essere incrementato quando il server sarà messo inproduzione.

```
/usr/local/sbin/clamav-milter --max-children=2 /var/run/clamav/clmilter.sock
```

faccio ripartire il sendmail

3.4 CONFIGURAZIONE

modifico il file di configurazione /etc/clamav.conf inserendo la riga per identificare il file di log

```
LogFile /var/log/clamav/clamd.log
```

creo la directory e file di log con le dovute proprietà

```
# mkdir /var/log/clamav/
# touch /var/log/clamav/clamd.log
# chown clamav /var/log/clamav/clamd.log
```

genero la directory di quarantena

```
# mkdir -p /var/mail/quarantine
# chown clamav:clamav /var/mail/quarantine
# chmod 700 /var/mail/quarantine
```

rifaccio partire il demone clamd

```
clamd
```

prima i fare rigenerare il socket, conviene rimuovere quello vecchio (clmilter.sock).

```
rm /var/run/clamav/clmilter.sock
```

aumento il numero di processi da 2 a 50

```
/usr/local/sbin/clamav-milter --max-children=50 --quarantine-dir=/var/mail/quarantine /var/run/clamav/clmilter.sock
```

Il file di log /var/log/clamav/clamd.log è istruttivo, questo è un file di esempio:

```
WARNING: Socket file /var/run/clamav/clamd.sock exists. Unclean shutdown? Removing...
Unix socket file /var/run/clamav/clamd.sock
Setting connection queue length to 15
ERROR: Can't save PID in file /var/run/clamd.pid
Archive: Archived file size limit set to 15728640 bytes.
Archive: Recursion level limit set to 8.
Archive: Files limit set to 1000.
Archive: Compression ratio limit set to 250.
Archive support enabled.
Archive: RAR support disabled.
Portable Executable support enabled.
Mail files support enabled.
OLE2 support enabled.
HTML support enabled.
Self checking every 1800 seconds.
```

3.5 STARTUP

genero uno scrip di startup

```
vi /etc/rc.d/clamd
```

e scrivo:

```
#! /bin/sh
#  Antivirus Clamav Startup (by A.Cora)
```

```
case "$1" in
start)
    /usr/local/sbin/clamd -c /etc/clamd.conf
    echo -n " clamd started"
    /usr/local/sbin/clamav-milter --max-children=50 --quarantine-dir=/var/mail/quarantine /
var/run/clamav/clmilter.sock
    echo -n " clmilter started"
    echo -n " "
     ;;
stop)
    killall clamd > /dev/null 2>&1
    killall /usr/local/sbin/clamav-milter > /dev/null
    rm /var/run/clamav/clamd.sock
    rm /var/run/clamav/clmilter.sock
    ;;
*)
    echo ""
    echo "Usage: clamd [start|stop]"
    echo ""
    exit
    ;;
esac
```

definisco le proprietà dello script:

```
chmod 700 clamd
```

installo lo script nella directory /etc/init.d e i suoi link nelle directory rc.3,rc.4,rc.5

3.6 AGGIORNAMENTO AUTOMATICO ANTIVIRUS

Visto il continuo sviluppo di codici maligni (virus) è indispensabile un sistema di aggiornamento automatico del datase dei virus, questa funzione è assolta dal tool freshclam incluso nel package di CLAMAV. Modifico il file di configrazione /etc/freshclam.conf, per l'utilizzo del proxy dell'osservatorio. Genero il file di log con le dovute proprieta-

```
# touch /var/log/freshclam.log
# chown clamav /var/log/freshclam.log
```

La directory del database

```
# mkdir /var/lib/clamav
# chown clamav /var/lib/clamav
```

Provo a lanciare il tool di aggiornamento per verificarne la funzionalità:

```
# freshclam
```

ottengo:

```
ClamAV update process started at Tue Apr 11 16:51:43 2006
Connecting via proxy.oato.inaf.it
Downloading main.cvd [*]
main.cvd updated (version: 37, sigs: 46700, f-level: 7, builder: ccordes)
Connecting via proxy.oato.inaf.it
Downloading daily.cvd [*]
daily.cvd updated (version: 1391, sigs: 3857, f-level: 7, builder: ccordes)
Database updated (50557 signatures) from db.it.clamav.net
```

modifico anche il file di configurazione principale clamad.conf che deve riconoscere a directory con gli aggiornamenti:

```
DatabaseDirectory /var/lib/clamav
```

l'aggiornamento dell'antivirs avviene quotidianamente tramite il demone crond, alle ore 3:09 di ogni primo del mese:

```
# crontab -l
9 3 1 1-12 * /usr/local/bin/freshclam --quiet
```

4.0 IL FILTRO ANTISPAM (SPAMASSASSIN)

La gestione dello spam presenta 2 aspetti fondamentali:

- evitare che i sistemi di istituto, ed in particolare il mailserver, sia utulizzato per inviare spam;
- ridurre al minimo lo spam ricevuto.

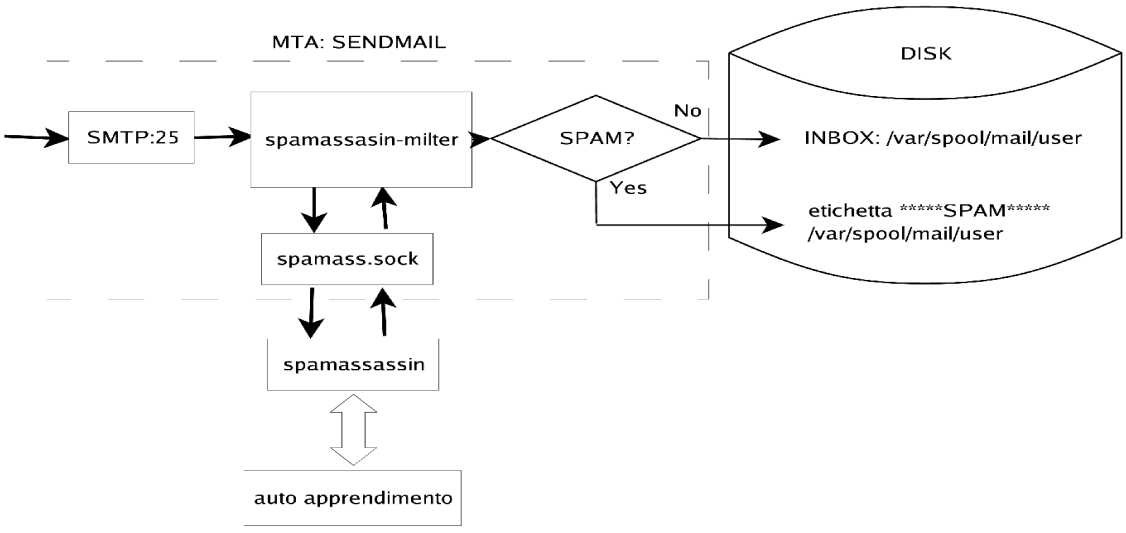

Figura 4: diagramma a blocchi dell'implementazione del filtro antispam

Per quanto riguarda il secondo punto, anche i MUA possono assolvere efficace ruolo antispam, sebbene un

antispam a livello di server è molto più efficiente ed efficace.
Per quanto concerne il primo punto, può essere realizzato solo a livello di server ed è essenziale per l'isttuto. Fino algli anni '90, quando internet non era usata per fini commerciali, era normale che i server SMTP fossero "open relay" e inoltrassero posta di qualsiasi provenienza. Quando il commercio di massa si impossesso dello strumento di spam , i mail server cessarono di essere aperti a chiunque, gli "open relay" scesero dal 90% all'1%.

Molti Internet Provider, utilizzano delle black list, liste di indirizzi IP da cui rifiutare posta al fine di ridurre il fenomeno di spam. Al fine di garantire la consegna della posta da arte dell'utenza dell'Osservatorio è quindi indispensable non rientrare in queste black list.

4.1 INSTALLAZIONE

Scarico la relase Mail-SpamAssassin-3.1.1 dal sito: http:// .....scompatto e compilo

```
cd Mail-SpamAssassin-3.1.1
```

SPAMASSASIN è un codice sviluppato in perl e come previsto da questo linguaggio:

```
perl Makefile.PL
```

attenzione, mancano dei moduli PERL, come evidenza il primo tentativo di complazione:

```
The "sa-update" script requires this module to access compressed
  update archive files.

optional module missing: Mail::SPF::Query
optional module missing: IP::Country
optional module missing: Razor2
optional module missing: Net::Ident
optional module missing: IO::Socket::INET6
optional module missing: IO::Socket::SSL
optional module missing: IO::Zlib
```

questi pacchetti si possono scaricare dall'archivio CPAN e installarli con la normale sintassi dei moduli PERL (perl Makefile.PL, make, make test, meke install). Completo l'installazione di SPAMASSASIN e definisco uno script per far partire il demone al boot.

```
# cp redhat-rc-script.sh /etc/init.d/spamd
```

Genero dei link software con i link S80spamassassin in rc5.d che identifica il demone /usr/bin/spamd. Una configurazione di base di spamassasin si può ottenere tramite il sito: http://www.yrex.com/spam/spamconfig.php che mi consente una configurazione iniziale di spamassasin. Il file di configurazione dell'antispam è posto nella directory /etc/sysconfig/spamassassin (/etc/mail/spamassassin) con nome local.cf e pubblicato in appendice. Effettuo il reboot e controllo che il demone sia attivato

```
[alberto@sam rc5.d]$ ps -eaf | grep spam
root     11250     1  0 Apr10 ?        00:00:00 /usr/bin/spamd -d -c -m5 -H -r /var/run/spamd.pid
```

anche in questo caso devo scaricare dalla rete delle librerie di spammassasin che servono ad integrarlo con il MILTER di SENDMAIL, scarico spamass-milter-0.3.1.tar.gz dal sito http://download.savannah.nongnu.org/releases/spamass-milt/, scompatto ed installo:

```
tar xvf spamass-milter-0.3.1.tar.gz
cd spamass-milter-0.3.1
./configure
make
make install
```

collego l'eseguibile dei milter per spamassasin,

```
 ln -s /usr/local/sbin/spamass-milter /usr/sbin/spamass-milter
```

Anche i milter di spammassasin sono gestiti con demoni da far partire al boot, trovo in una sottodirectory delle librerie, lo script necessario allo startup:

```
cd contrib
cp spamass-milter-redhat.rc /etc/init.d/spamass-milter
chmod 700 /etc/init.d/spamass-milter
```

Provo lo script:

```
root@sam spamass-milter-0.3.1]# /etc/init.d/spamass-milter start
Starting spamass-milter:                                   [  OK  ]
```

Collego lo script nell'opportuna directory, attivandoli prima di SENDMAIL:

```
cd /etc/rc5.d
ln -s /etc/init.d/spamass-milter S80spamass-milter
```

Il comando genera un socket, visibile come file:

```
ls /var/run/spamass.sock
```

4.2 CONFIGURAZIONE

Aggiorno la configurazione di sendmail, modificando il file /etc/mal/sendmail.mc ove inserisco le segueti righe:

```
dnl #
dnl # enable SpamAssassin (by A.Cora)
dnl #
```

```
INPUT_MAIL_FILTER(`spamassassin', `S=local:/var/run/spamass.sock, F=,T=C:15m;S:4m;R:4m;E:10m')
dnl #
```

ricompilo il file di configurazione di sendmail

```
# make -C /etc/mail
make: Entering directory `/etc/mail'
make: Leaving directory `/etc/mail'
```

faccio ripartire il servizio:

```
# /etc/init.d/sendmail restart
Shutting down sendmail:                    [  OK  ]
Shutting down sm-client:                   [  OK  ]
Starting sendmail:                         [  OK  ]
Starting sm-client:                        [  OK  ]
```

Un'alternativa per far ripartire il servizio SENDMAIL è il comando kill che manda un SIGHUP per far ripartire e il processo leggendo la nuova configurazione del sendmail.

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

4.3 PERSONALIZZAZIONE FILTRO ANTISPAM

Il filtraggio automatico dei mail indesiderati è realizzato a livello di server di posta da SpamAssassin. La personalizzazione del filtro avviene tramite il programma procmail, gia presente nella distribuzione di Linux installata.
Procmail è un programma che consente di intercettare la posta in arrivo prima che venga depositata nella mailbox dell'utente. Lo scopo è quello di applicare a ciascun messaggio in arrivo delle regole definite dall'utente stesso tramite uno o più file di configurazione.
Nel caso di operazioni di filtraggio particolarmente complesse, come il riconoscimento degli spam, procmail può essere configurato in modo da passare il controllo ad un programma specializzato, nel nostro caso SpamAssassin.
Per portare a termine il suo compito SpamAssassin utilizza un gran numero di regole "euristiche": ogni test origina un punteggio positivo (possibile spam) o negativo (possibile messaggio regolare) o nullo, e alla fine la somma algebrica viene confrontata con un valore di soglia (il default è 5). Se il risultato finale supera la soglia il messaggio viene considerato spam.
SpamAssassin configurato nel server di posta non elimina gli spam, si limita a marcarli con una stringa di riconoscimento per poi passarli all'utente (che può decidere di elaborarli con procmail). A questo punto procmail può applicare ai messaggi ulteriori regole definite nei file .procmailrc e .spamassassin/user_prefs personalizzabili dall'utente.

L'attivazione di procmail che definisce i settaggi personalizzati deve essere compiuta` dall'utente stesso, accedendo al proprio account sul mail server

```
ssh -l [nomeutente] sam.oato.inaf.it
```

Creare un file di nome .procmailrc copiandone uno gia` esistente:

```
cp /home/alberto/.procmailrc .
```

che serve a configurare procmail e contiene le seguenti istruzioni

```
#----------------------
DEFAULT=/var/spool/mail/$USER
VERBOSE=on
PMDIR=$HOME/.procmail
LOGFILE=$PMDIR/log
#
# Invoke SpamAssassin
#
:0fw
| /usr/bin/spamassassin --prefs-file=$HOME/.spamassassin/user_prefs
```

Le regola e' una sola , introdotta dalla sequenza :0, (per una descrizione dettagliata della sintassi utilizzata in queste regole o aggiungerne di nuove vedere il comando man procmailrc.), e invoca l'utilizzo del file di personalizzazione di spamassasin.
Creare la directory e il file di log:

```
mkdir .procmail
touch .procmail/log
```

Creare la directory e il file con le regole personali di spamassassin.

```
mkdir .spamassassin
```

Anche questo file può essere, per comodità, copiato questo file puo' essere copiato:

```
cp /home/alberto/.spamassassin/user_prefs .spamassassin/.
```

Ovviamente le regole che possono essere modificate e affinate dall'utente. Nella configurazione di default alla data odierna sono:

```
# abbassa la soglia del punteggio da 5 a 2
# per classificare il messaggio USPAM
required_score 2.00
# aggiunge al soggetto del mail la scriita *****USPAM*****
rewrite_header Subject *****USPAM*****
# whitelist_from
whitelist_from *@*.inaf.it #friend@somewhere.com *@isp.com *.domain.net
```

```
# blacklist_from
blacklist_from *@spammer.com
```

In particolare si possono definire blacklist e whitelist e modificare il valore di soglia che discrimina gli spam. Attenzione, abbiamo posto nella whitelist qualsiasi email apparentemente spedita dall'INAF, significa che gli spammer che si cammuffano con questo dominio non verranno intercettati.

5.0 LOCAL DELIVERY AGENT

L'accesso locale ai messaggi di posta elettronica (LDA) oltre ad un accesso diretto sull'account, si basa su 2 protocolli (POP3 e IMAP4) alquanto datati che non prendono nella dovuta consiferazione i problemi di sicurezza. In particolare nella loro implementazione base, questi protocolli soffrono di 2 problemi:

- il traffico, sia messaggi che password, avviene "in chiaro", cioè è possibile per un malintenzionato, dotato di opportuni strumenti, leggerne il contenuto e impossessarsi dell'account.
- non è disponibile un meccanismo di identificazione certa del sistema di posta, consentendo al solito maleintenzionato di simulare il server di posta elettronica.

In realtà il rischio nell'uso di questi servizi non criptati all'interno dell'Osservatorio è minimo, grazie alla presenza dl firewall; invece il problema emerge in modo drammatico per gli utenti che si desiderano collegare dall'esterno, utilizzando computer portatili o connessioni casalinghe.
Per questi usi occasionali, è stato predisposto una consultazione via WEB tramite Mailman (con protocollo sicuro HTTPS) oppure i protocolli POPS e IMAPS che accedono alle porte (995 e 993) tramite il supporto TLS/SSL. Si è adottato quale software per distribuire la posta elettronica DOVECOD che è incluso nella distribuzione.
DOVECOD supporta i protocolli POP e IMAP e le loro estensioni sicure POPS e IMAPS.

5.1 CONFIGURAZIONE (DOVECOD)

Visto che il servizio, deve supportare protocolli di sicurezza, inizio compilando il certificato e il suo file di configurazione:
/etc/pki/dovecot/dovecot-openssl.cnf. Per generare il certificato :

```
cd /etc/pki/dovecot/
mv dovecot.pem dovecot.pem.old
mv private/dovecot.pem private/dovecot.pem.old

/usr/share/doc/dovecot-0.99.14/examples/mkcert.sh
/etc/pki/dovecot/certs directory doesn't exist
Generating a 1024 bit RSA private key
.....++++++
.....++++++
writing new private key to '/etc/pki/dovecot/private/dovecot.pem'
-----

subject= /C=IT/ST=ITALY/L=Pino Torinese/O=INAF-OATo/OU=IMAP server
```

OATo/CN=oato.inaf.it/emailAddress=postmaster@oato.inaf.it
MD5 Fingerprint=D2:5E:17:E2:19:12:1F:62:C6:4B:05:ED:92:8B:D2:05

Il software DOVECOT presenta già gli opportuni script per il suo startup.

6.0 INSTALLAZIONE SOFWARE COMPLEMENTARE

Al servizio di posta elettronica si aggiungono un piccolo kit di softare complementare, che può essere utile all'utente del servizio: PINE, VACATION e MESSAGGIO DI BENVENUTO.

6.1 PINE

Pine è un MUA che utilizza un interfaccia con caratteri testo e che può essere utilizzato collegandosi via ssh al server di posta. Scarico il software gia compilato per redhat e quindi anche per la nostra distribuzione Linux (*.rpm) e rigenero i link alle libreie

```
rpm -i pine-bin.linux-rhe3.rpm
```

genero manualmente i link:

```
ln -s /usr/lib/libldap-2.2.so.7.0.16 /usr/lib/libldap.so.
ln -s /lib/libssl.so.0.9.7f /usr/lib/libssl.so.4
ln -s /lib/libcrypto.so.4 /lib/libcrypto.so.0.9.7f
```

rinomino file /bin/pine

6.2 VACATION

VACATION è un software che consente di inviare un messaggio di risposta automatico alla ricezione di un mail (funzione egreggiamente svolta anche da Procmail); questa capacità è anche implementata da molti MUA, però ci senbra conveniente di dotarci a livello di server di posta di questa funzionalità in modo da non essere obbligati a tenere coputer client accesi nei periodi di vacanze.
Per l'utilizzo di VACATION, si legga il manuale disponibile in linea (man vacation). L'utility VACATION e' normalmente inclusa con SENDMAIL, putroppo come nel caso delle LIBMILTER non sembra essere immediatamente disponibile, per questa ragione compilo i sorgenti che ho gia ottenuto scaricando il software e compilamdolo:

```
cd sendmail-8.13.6/vacation
```

```
make
```

```
make install
```

6.3 MESSAGGIO DI BENVENUTO

Gli utenti che si collegano al servizio di posta elettronica sono accolti da un messaggio di benvenuto, utile a ricordare l'email a cui possono rivolgersi per eventuali problemi. Per generare il messaggio è sufficiente aggiungere nella directory /etc il file motd, che nel nostro caso contiene il seguente testo

```
** Benvenuti nel servizio di posta elettronica INAF-OATo **
       ** Welcome to INAF-OATo e-mail service **
_____....____
           Support: supporto@oato.inaf.it
```

7.0 HARDENING

La sicurezza del servizio deve essere garantita da un oppropriato hardening del server. Inanzitutto il server è mono servizio, per questa ragione sono accessibili porte finalizzate a proptocolli propri del servizio di posta. Per esaminare le porte del server:

```
nmap -P0 sam.oato.inaf.it
```

Alla riduzione delle porte e dei servizi disponibili si aggiungono altre operazioni di hardening**,** quali il tunnel per un accesso protetto alla porta 25 del protocollo SMTP dall'esterno e l'installazione di una shell ridotta (rbash), che lmita i movimenti dell'utente che accede via ssh.

7.1 TUNNEL SICURO PER SMTPS

Scarico il software stunnel dal sito: http://.www.stunnel.org

Dopo che ho acquisito i privilegi di root, genero il gruppo nogroup (groupadd), decomprimo il pacchetto e compilo:

```
./configure
make
make install
```

... durante questa fase, il software richiede la compilazione dei certificati:

```
Generating a 1024 bit RSA private key
....++++++
.............++++++
writing new private key to 'stunnel.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
```

```
Country Name (2 letter code) [PL]:IT
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:Pino Torinese
Organization Name (eg, company) [Stunnel Developers Ltd]:INAF-OATo
Organizational Unit Name (eg, section) []:CED
Common Name (FQDN of your server) [localhost]:sam.oato.inaf.it
subject= /C=IT/ST=Some-State/L=Pino Torinese/O=INAF-OATo/OU=CED/CN=sam.oato.inaf.it
notBefore=Apr 26 12:22:00 2006 GMT
notAfter=Apr 26 12:22:00 2007 GMT
MD5 Fingerprint=BF:18:5A:FD:29:57:8A:4B:02:D8:82:52:E6:EE:E3:08
/usr/bin/install -c -d -m 1770 -g nogroup /usr/local/var/lib/stunnel
if uname | grep SunOS; then \
        /usr/bin/install -c -d -m 755 /usr/local/var/lib/stunnel/dev; \
        mknod /usr/local/var/lib/stunnel/dev/zero c 13 12; \
        chmod 666 /usr/local/var/lib/stunnel/dev/zero; \
fi
test -z "/usr/local/share/doc/stunnel/examples" || mkdir -p -- "/usr/local/share/doc/stunnel/examples"
 /usr/bin/install -c -m 644 'ca.html' '/usr/local/share/doc/stunnel/examples/ca.html'
 /usr/bin/install -c -m 644 'ca.pl' '/usr/local/share/doc/stunnel/examples/ca.pl'
 /usr/bin/install -c -m 644 'importCA.html' '/usr/local/share/doc/stunnel/examples/importCA.html'
 /usr/bin/install -c -m 644 'importCA.sh' '/usr/local/share/doc/stunnel/examples/importCA.sh'
 /usr/bin/install -c -m 644 'script.sh' '/usr/local/share/doc/stunnel/examples/script.sh'
 /usr/bin/install -c -m 644 'stunnel.spec' '/usr/local/share/doc/stunnel/examples/stunnel.spec'
 /usr/bin/install -c -m 644 'stunnel.init' '/usr/local/share/doc/stunnel/examples/stunnel.init'
make[2]: Leaving directory `/home/alberto/STUNNEL/stunnel-4.15/tools'
make[1]: Leaving directory `/home/alberto/STUNNEL/stunnel-4.15/tools'
make[1]: Entering directory `/home/alberto/STUNNEL/stunnel-4.15'
make[2]: Entering directory `/home/alberto/STUNNEL/stunnel-4.15'
make[2]: Nothing to be done for `install-exec-am'.
test -z "/usr/local/share/doc/stunnel" || mkdir -p -- "/usr/local/share/doc/stunnel"
 /usr/bin/install -c -m 644 'AUTHORS' '/usr/local/share/doc/stunnel/AUTHORS'
 /usr/bin/install -c -m 644 'BUGS' '/usr/local/share/doc/stunnel/BUGS'
 /usr/bin/install -c -m 644 'ChangeLog' '/usr/local/share/doc/stunnel/ChangeLog' /usr/bin/install -c -m 644 'COPYING'
'/usr/local/share/doc/stunnel/COPYING'
 /usr/bin/install -c -m 644 'COPYRIGHT.GPL' '/usr/local/share/doc/stunnel/COPYRIGHT.GPL'
 /usr/bin/install -c -m 644 'CREDITS' '/usr/local/share/doc/stunnel/CREDITS'
 /usr/bin/install -c -m 644 'INSTALL' '/usr/local/share/doc/stunnel/INSTALL'
 /usr/bin/install -c -m 644 'INSTALL.W32' '/usr/local/share/doc/stunnel/INSTALL.W32'
 /usr/bin/install -c -m 644 'INSTALL.WCE' '/usr/local/share/doc/stunnel/INSTALL.WCE'
 /usr/bin/install -c -m 644 'NEWS' '/usr/local/share/doc/stunnel/NEWS'
 /usr/bin/install -c -m 644 'PORTS' '/usr/local/share/doc/stunnel/PORTS'
 /usr/bin/install -c -m 644 'README' '/usr/local/share/doc/stunnel/README'
 /usr/bin/install -c -m 644 'TODO' '/usr/local/share/doc/stunnel/TODO'
make[2]: Leaving directory `/home/alberto/STUNNEL/stunnel-4.15'
make[1]: Leaving directory `/home/alberto/STUNNEL/stunnel-4.15'
```

A differenza di SENDMAIL, SPAMASSASIN e CLAMAV che possiedo uno startup tramite la drectory init.dgenero il file / della configurazione originaria di Berkely, il tunnel lo attiviamo tramite XINETD e quindi mi genero un file, che normalmente viene incluso allo startup di XINETD e descrive il servizio : /etc/xinetd.d/smtps

```
service smtps
{
disable = no
socket_type = stream
wait = no
user = root
server = /usr/sbin/stunnel
server_args = -v3 -rlocalhost:25
log_on_success += HOST DURATION
log_on_failure += HOST
}
```

faccio ripartire xinetd

```
/sbin/service xinetd restart
```

7.3 RBASH

La shell bash distribuia con il nostro sistema operativa, è dotata di una riduzione dei comandi che si ottiene digitando rbash (oppure bash -r), mi genero /bin/rbash tramite un link software:

```
ln -s /bin/bash /bin/rbash
```

che utilizzero nel generare le utenze

8.0 MIGRAZIONE

L'installazione del server di posta dell'Osservatorio Astronomico di Torino, è stata lunga e complicata,per una ragione molto semplice: piena compatibilità con il server postino precedente, a questo si aggiunge il fatto che per ragioni di continuita del servizio i due server devono coesistere fino a migrazione ultimata. Conclusa l'installazione, descrivaimo qui di seguito la procedura di migrazione delle utenze.

8.1 GENERAZIONE DELLE UTENZE

Definisco il gruppo users gid:100 (users) che sarà utilzzato da tutti gli utenti mi genero le utenze tramite lo script utente [nome] [uid] che riporto in appendice. Lo script è richiamato da un secondo codice ove uid e guid sono identici a quelli utilizzati dal vecchio servizio di posta, nominato fai_utenti.

```
# chmod 700 fai_utenti
```

./fai_utenti

In questo modo abbiamo generatoi 93 utenti già presenti nel vechio servizio, ogni utente ha un file .forward che rimanderà la posta nei vecchi mail box su zeus.to.astro.it, e quindi la coesistenza di due server mail all'interno del dominio. Segue l'aggiornamento del file /etc/aliases, che deve contenere la corrispondenza tra nome dell'utente e destinazione. Dopo aver modificato il file, aggiorno la lista con il comando newaliases

newaliases
/etc/aliases: 182 aliases, longest 580 bytes, 3818 bytes total

8.2 MIGRAZIONE DELLE CASELLE DI POSTA

Fisicamente, i file che contengono i messaggi sono memrizzati nella directory /var/spool/mail e nella directory /home/users come nel vecchio server postino. Dal lato server esistono 3 possibili casi per l'utilizzo del server:

- l'utente utilizza l'account tramite un forward che rimanda la posta su una macchina client. con .forward Al fine di far transitare queto titpo di utenza sarebbe sufficiente aggiornare il .forward nella directory dell'utente.
- l'utente utilizza pop, sarebbe sufficiente copiare la casela di posta in /var/spool/mail
- l'utente utilizza imap, dobbiamo copiare la casela di posta in /var/spool/mail e l'area della account che si trova nella directory /home/user

La scelta è di realizzare uno script che soddisfi il terzo caso e quindi anche i primi due casi meno complicati.
Condivido via NFS le directory conteneti gli e-mail dal nuovo server (sam.oato.inaf.it) al vecchio servr (zeus.to.astro.it), e le monto sul vecchio server:

mkdir /migrazione/spool
mkdir /migrazione/home
mount sam.oato.inaf.it:/var/spool/mail /migrazione/spool
mount sam:/home/users /migrazione/home

Sul vecchio server preparo lo script sposta_la_posta (vedi appendice), che mi copia l'area dell'utente e la casella di posta sulle directory condivise. La presenza di un .forward su zeus che rimanda i mail sul nuovo sever ci consente la coesistena dei due server diposta fino all'unificazione dell MX record.

9.0 ASSISTENZA E MANUTENZIONE

Per comodità qui di seguito, ripetiamo i vari comandi per stato|riavvio|fermata|avvio dei vari servizi:

MTA: sendmail
/etc/init.d/sendmail status|restart | stop | start

Filtri Antivirus sono configurati e azionati trmite sendmail

Filtri Antispam
/etc/init.d/spamassassin status|restart |stop | start
/etc/init.d/spamass-milter status|restart |stop | start

LDA: Dovecot
/etc/init.d/dovecot status|restart | stop | start

SMTPS tramite stunnel viene attivato da xinetd
/sbin/service xinetd restart

9.1 SPAMM-MILTER

Mentre l'antivirus clamAV e il suo milter filtrano senza cedimenti i mail, il filtro antispamm talvolta si blocca lasciando passare messaggi senza filtrarli.

Non avendo compreso la causa di questo anomalo comportamento ho scritto un piccolo codiche in bash/awhk / root/script/controlla_spam-milter-sock.sh che viene gestito dal crontab e verifica la funzionalita` del servizio ogni ora.

9.2 SPAZIO DISCO

Si e` realizzato lo script /root/script/controlla_spazio_disco.sh che verifica la dispobilita` di memoria al fine di evitare il crash del server. Lo script e` gestito dal crontab e verifica che la posta sia inferiore a 200GB. In caso negativo manda un warning via e-mail

9.3 LOG REPPOSITORY

Per semplificare l'assistenza e manutenzione del server si e` scelto di realizzare un collegamento con i vari log del sistema in una sotto directory /root/LOG (nota bene la directory /root/LOG/VAR è un link software della directory / var/log).
Per comodità riepiloghiamo la posizione dei vari file di log:

Funzionalita' dello spam-milter-sock : /root/LOG/spamm-milter.log
Tutto quanto è gestito dall'MTA : /root/LOG/VAR/maillog
Dovecot : /root/LOG/VAR/dovecod.log
Aggiornamenti Antivirus Clamav : /root/LOG/VAR/freshclam.log
Antivirus Clamav : /root/LOG/VAR/clamav.log
Demone Clamd : /root/LOG/VAR/clamav/clamd.log
Accessi via ssh : /root/LOG/VAR/secure

repository dei mail con i virus : /var/mail/quarantine

10.0 CONCLUSIONI

L'operazione di installazione, configurazione e migrazioni delle utenze dell'Osservatorio Astronomico ha richiesto dal Marzo ad Ottobre 2006, senza per questo interrompere il servizio di posta elettronica.

Questo documento condensa le operazioni compiute ed e` stato gia utilizzato per svolgere attivita` di screening e assistenza al sistema medesimo

APPENDICE A: sendmail.mc e file ausiliari

```
divert(-1)dnl
dnl #
dnl # This is the sendmail macro config file for m4. If you make changes to
dnl # /etc/mail/sendmail.mc, you will need to regenerate the
dnl # /etc/mail/sendmail.cf file by confirming that the sendmail-cf package is
dnl # installed and then performing a
dnl #
dnl #     make -C /etc/mail
dnl #
include(`/usr/share/sendmail-cf/m4/cf.m4')dnl
VERSIONID(`setup for Red Hat Linux')dnl
OSTYPE(`linux')dnl
dnl #
dnl # default logging level is 9, you might want to set it higher to
dnl # debug the configuration
dnl #
dnl define(`confLOG_LEVEL', `9')dnl
dnl #
dnl # Uncomment and edit the following line if your outgoing mail needs to
dnl # be sent out through an external mail server:
dnl #
dnl #define(`SMART_HOST',`smtp.your.provider')
dnl #
define(`confDEF_USER_ID',``8:12'')dnl
dnl define(`confAUTO_REBUILD')dnl
define(`confTO_CONNECT', `1m')dnl
define(`confTRY_NULL_MX_LIST',true)dnl
define(`confDONT_PROBE_INTERFACES',true)dnl
define(`PROCMAIL_MAILER_PATH',`/usr/bin/procmail')dnl
define(`ALIAS_FILE', `/etc/aliases')dnl
define(`STATUS_FILE', `/var/log/mail/statistics')dnl
define(`UUCP_MAILER_MAX', `2000000')dnl
define(`confUSERDB_SPEC', `/etc/mail/userdb.db')dnl
define(`confPRIVACY_FLAGS', `authwarnings,novrfy,noexpn,restrictqrun')dnl
define(`confAUTH_OPTIONS', `A')dnl
dnl #
dnl # The following allows relaying if the user authenticates, and disallows
dnl # plaintext authentication (PLAIN/LOGIN) on non-TLS links
dnl #
dnl define(`confAUTH_OPTIONS', `A p y')dnl
dnl #
dnl # PLAIN is the preferred plaintext authentication method and used by
dnl # Mozilla Mail and Evolution, though Outlook Express and other MUAs do
dnl # use LOGIN. Other mechanisms should be used if the connection is not
dnl # guaranteed secure.
dnl # Please remember that saslauthd needs to be running for AUTH.
dnl #
TRUST_AUTH_MECH(`EXTERNAL DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
define(`confAUTH_MECHANISMS', `EXTERNAL GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
dnl #
dnl # Rudimentary information on creating certificates for sendmail TLS:
dnl #     cd /usr/share/ssl/certs; make sendmail.pem
dnl # Complete usage:
dnl #     make -C /usr/share/ssl/certs usage
dnl #
define(`confCACERT_PATH',`/etc/pki/tls/certs')
define(`confCACERT',`/etc/pki/tls/certs/ca-bundle.crt')
define(`confSERVER_CERT',`/etc/pki/tls/certs/sendmail.pem')
define(`confSERVER_KEY',`/etc/pki/tls/certs/sendmail.pem')
dnl #define(`confCLIENT_CERT',`/etc/pki/tls/certs/sendmail.pem')
dnl #define(`confCLIENT_KEY',``/etc/pki/tls/certs/sendmail.pem')
```

```
dnl #
dnl # This allows sendmail to use a keyfile that is shared with OpenLDAP's
dnl # slapd, which requires the file to be readble by group ldap
dnl #
dnl define(`confDONT_BLAME_SENDMAIL',`groupreadablekeyfile')dnl
dnl #
dnl define(`confTO_QUEUEWARN', `4h')dnl
dnl define(`confTO_QUEUERETURN', `5d')dnl
dnl define(`confQUEUE_LA', `12')dnl
dnl define(`confREFUSE_LA', `18')dnl
define(`confTO_IDENT', `0')dnl
dnl FEATURE(delay_checks)dnl
FEATURE(`no_default_msa',`dnl')dnl
FEATURE(`smrsh',`/usr/sbin/smrsh')dnl
FEATURE(`mailertable',`hash -o /etc/mail/mailertable.db')dnl
FEATURE(`virtusertable',`hash -o /etc/mail/virtusertable.db')dnl
FEATURE(redirect)dnl
FEATURE(always_add_domain)dnl
FEATURE(use_cw_file)dnl
FEATURE(use_ct_file)dnl
dnl #
dnl # The following limits the number of processes sendmail can fork to accept
dnl # incoming messages or process its message queues to 12.) sendmail refuses
dnl # to accept connections once it has reached its quota of child processes.
dnl #
dnl define(`confMAX_DAEMON_CHILDREN', 60)dnl
dnl #
dnl # Limits the number of new connections per second. This caps the overhead
dnl # incurred due to forking new sendmail processes. May be useful against
dnl # DoS attacks or barrages of spam. (As mentioned below, a per-IP address
dnl # limit would be useful but is not available as an option at this writing.)
dnl #
dnl define(`confCONNECTION_RATE_THROTTLE', :30)dnl
dnl #
dnl # The -t option will retry delivery if e.g. the user runs over his quota.
dnl #
FEATURE(local_procmail,``,`procmail -t -Y -a $h -d $u')dnl
FEATURE(`access_db',`hash -T<TMPF> -o /etc/mail/access.db')dnl
FEATURE(`blacklist_recipients')dnl
EXPOSED_USER(`root')dnl
dnl #
dnl # The following causes sendmail to only listen on the IPv4 loopback address
dnl # 127.0.0.1 and not on any other network devices. Remove the loopback
dnl # address restriction to accept email from the internet or intranet.
dnl #
DAEMON_OPTIONS(`Port=25, Name=MTA')dnl
dnl #
dnl # The following causes sendmail to additionally listen to port 587 for
dnl # mail from MUAs that authenticate. Roaming users who can't reach their
dnl # preferred sendmail daemon due to port 25 being blocked or redirected find
dnl # this useful.
dnl #
dnl # DAEMON_OPTIONS(`Port=submission, Name=MSA, M=Ea')dnl
dnl #
dnl # The following causes sendmail to additionally listen to port 465, but
dnl # starting immediately in TLS mode upon connecting. Port 25 or 587 followed
dnl # by STARTTLS is preferred, but roaming clients using Outlook Express can't
dnl # do STARTTLS on ports other than 25. Mozilla Mail can ONLY use STARTTLS
dnl # and doesn't support the deprecated smtps; Evolution <1.1.1 uses smtps
dnl # when SSL is enabled-- STARTTLS support is available in version 1.1.1.
dnl #
dnl # For this to work your OpenSSL certificates must be configured.
dnl #
dnl #DAEMON_OPTIONS(`Port=smtps, Name=TLSMTA, M=s')dnl
dnl #
dnl # The following causes sendmail to additionally listen on the IPv6 loopback
```

```
dnl # device. Remove the loopback address restriction listen to the network.
dnl #
dnl DAEMON_OPTIONS(`port=smtp,Addr=::1, Name=MTA-v6, Family=inet6')dnl
dnl #
dnl # enable both ipv6 and ipv4 in sendmail:
dnl #
dnl DAEMON_OPTIONS(`Name=MTA-v4, Family=inet, Name=MTA-v6, Family=inet6')
dnl #
dnl # enable claamav antivirus (by A.Cora)
dnl #
INPUT_MAIL_FILTER(`clamav', `S=local:/var/run/clamav/clmilter.sock, F=, T=S:4m;R:4m')dnl
define(`confINPUT_MAIL_FILTERS', `clamav')
dnl #
dnl #
dnl # enable SpamAssassin (by A.Cora)
dnl #
INPUT_MAIL_FILTER(`spamassassin', `S=local:/var/run/spamass.sock, F=,T=C:15m;S:4m;R:4m;E:10m')
dnl #
dnl # enable forward (by A.Cora)
dnl #
OPTION ('confDONT_BLAME_SENDMAIL', 'forwardfileingroupwritabledirpath')dnl
dnl #
dnl # We strongly recommend not accepting unresolvable domains if you want to
dnl # protect yourself from spam. However, the laptop and users on computers
dnl # that do not have 24x7 DNS do need this.
dnl #
FEATURE(`accept_unresolvable_domains')dnl
dnl #
FEATURE(`relay_based_on_MX')dnl
dnl #
dnl # Also accept email sent to "localhost.localdomain" as local email.
dnl #
LOCAL_DOMAIN(`sam.oato.inaf.it')dnl
dnl #
dnl # The following example makes mail from this host and any additional
dnl # specified domains appear to be sent from mydomain.com
dnl #
MASQUERADE_AS(`oato.inaf.it')dnl
dnl #
dnl # masquerade not just the headers, but the envelope as well
dnl #
FEATURE(masquerade_envelope)dnl
dnl #
dnl # masquerade not just @mydomainalias.com, but @*.mydomainalias.com as well
dnl #
FEATURE(masquerade_entire_domain)dnl
dnl #
MASQUERADE_DOMAIN(localhost)dnl
MASQUERADE_DOMAIN(localhost.localdomain)dnl
MASQUERADE_DOMAIN(to.astro.it)dnl
dnl MASQUERADE_DOMAIN(mydomain.lan)dnl
MAILER(smtp)dnl
MAILER(procmail)dnl
```

APPENDICE B: clamad.conf

```
##
## Example config file for the Clam AV daemon
## Please read the clamd.conf(5) manual before editing this file.
##


# Comment or remove the line below.
# Example

# Uncomment this option to enable logging.
# LogFile must be writable for the user running daemon.
# A full path is required.
# Default: disabled
LogFile /var/log/clamav/clamd.log

# By default the log file is locked for writing - the lock protects against
# running clamd multiple times (if you want to run another clamd instance,
# please # copy the configuration file, change the LogFile variable, and run
# the daemon with the --config-file option).
# This option disables log file locking.
# Default: disabled
#LogFileUnlock

# Maximal size of the log file.
# Value of 0 disables the limit.
# You may use 'M' or 'm' for megabytes (1M = 1m = 1048576 bytes)
# and 'K' or 'k' for kilobytes (1K = 1k = 1024 bytes). To specify the size
# in bytes just don't use modifiers.
# Default: 1M
LogFileMaxSize 2M

# Log time with each message.
# Default: disabled
#LogTime

# Also log clean files. Useful in debugging but drastically increases the
# log size.
# Default: disabled
#LogClean

# Use system logger (can work together with LogFile).
# Default: disabled
LogSyslog

# Specify the type of syslog messages - please refer to 'man syslog'
# for facility names.
# Default: LOG_LOCAL6
#LogFacility LOG_MAIL

# Enable verbose logging.
# Default: disabled
#LogVerbose

# This option allows you to save a process identifier of the listening
# daemon (main thread).
# Default: disabled
# PidFile /var/run/clamd.pid

# Optional path to the global temporary directory.
# Default: system specific (usually /tmp or /var/tmp).
```

```
#TemporaryDirectory /var/tmp

# Path to the database directory.
# Default: hardcoded (depends on installation options)
DatabaseDirectory /var/lib/clamav

# The daemon works in a local OR a network mode. Due to security reasons we
# recommend the local mode.

# Path to a local socket file the daemon will listen on.
# Default: disabled
LocalSocket /var/run/clamav/clamd.sock
# LocalSocket /tmp/clamd

# Remove stale socket after unclean shutdown.
# Default: disabled
FixStaleSocket

# TCP port address.
# Default: disabled
#TCPSocket 3310

# TCP address.
# By default we bind to INADDR_ANY, probably not wise.
# Enable the following to provide some degree of protection
# from the outside world.
# Default: disabled
#TCPAddr 127.0.0.1

# Maximum length the queue of pending connections may grow to.
# Default: 15
#MaxConnectionQueueLength 30

# Clamd uses FTP-like protocol to receive data from remote clients.
# If you are using clamav-milter to balance load between remote clamd daemons
# on firewall servers you may need to tune the options below.

# Close the connection when the data size limit is exceeded.
# The value should match your MTA's limit for a maximal attachment size.
# Default: 10M
StreamMaxLength 20M

# Limit port range.
# Default: 1024
#StreamMinPort 30000
# Default: 2048
#StreamMaxPort 32000

# Maximal number of threads running at the same time.
# Default: 10
#MaxThreads 20

# Waiting for data from a client socket will timeout after this time (seconds).
# Value of 0 disables the timeout.
# Default: 120
#ReadTimeout 300

# Waiting for a new job will timeout after this time (seconds).
# Default: 30
#IdleTimeout 60

# Maximal depth directories are scanned at.
# Default: 15
#MaxDirectoryRecursion 20

# Follow directory symlinks.
```

```
# Default: disabled
#FollowDirectorySymlinks

# Follow regular file symlinks.
# Default: disabled
#FollowFileSymlinks

# Perform internal sanity check (database integrity and freshness).
# Default: 1800 (30 min)
#SelfCheck 600

# Execute a command when virus is found. In the command string %v will
# be replaced by a virus name.
# Default: disabled
#VirusEvent /usr/local/bin/send_sms 123456789 "VIRUS ALERT: %v"

# Run as a selected user (clamd must be started by root).
# Default: disabled
User clamav

# Initialize supplementary group access (clamd must be started by root).
# Default: disabled
#AllowSupplementaryGroups

# Stop daemon when libclamav reports out of memory condition.
#ExitOnOOM

# Don't fork into background.
# Default: disabled
#Foreground

# Enable debug messages in libclamav.
# Default: disabled
#Debug

# Do not remove temporary files (for debug purposes).
# Default: disabled
#LeaveTemporaryFiles


# By default clamd uses scan options recommended by libclamav. This option
# disables recommended options and allows you to enable selected ones below.
# DO NOT TOUCH IT unless you know what you are doing.
# Default: disabled
#DisableDefaultScanOptions

##
## Executable files
##

# PE stands for Portable Executable - it's an executable file format used
# in all 32-bit versions of Windows operating systems. This option allows
# ClamAV to perform a deeper analysis of executable files and it's also
# required for decompression of popular executable packers such as UPX, FSG,
# and Petite.
# Default: enabled
#ScanPE

# With this option clamav will try to detect broken executables and mark
# them as Broken.Executable
# Default: disabled
#DetectBrokenExecutables


##
## Documents
```

```
##

# This option enables scanning of Microsoft Office document macros.
# Default: enabled
#ScanOLE2

##
## Mail files
##

# Enable internal e-mail scanner.
# Default: enabled
ScanMail

# If an email contains URLs ClamAV can download and scan them.
# WARNING: This option may open your system to a DoS attack.
#                Never use it on loaded servers.
# Default: disabled
#MailFollowURLs


##
## HTML
##

# Perform HTML normalisation and decryption of MS Script Encoder code.
# Default: enabled
#ScanHTML


##
## Archives
##

# ClamAV can scan within archives and compressed files.
# Default: enabled
ScanArchive

# Due to license issues libclamav does not support RAR 3.0 archives (only the
# old 2.0 format is supported). Because some users report stability problems
# with unrarlib it's disabled by default and you must uncomment the directive
# below to enable RAR 2.0 support.
# Default: disabled
#ScanRAR

# The options below protect your system against Denial of Service attacks
# using archive bombs.

# Files in archives larger than this limit won't be scanned.
# Value of 0 disables the limit.
# Default: 10M
ArchiveMaxFileSize 15M

# Nested archives are scanned recursively, e.g. if a Zip archive contains a RAR
# file, all files within it will also be scanned. This options specifies how
# deep the process should be continued.
# Value of 0 disables the limit.
# Default: 8
#ArchiveMaxRecursion 9

# Number of files to be scanned within an archive.
# Value of 0 disables the limit.
# Default: 1000
#ArchiveMaxFiles 1500

# If a file in an archive is compressed more than ArchiveMaxCompressionRatio
```

```
# times it will be marked as a virus (Oversized.ArchiveType, e.g. Oversized.Zip)
# Value of 0 disables the limit.
# Default: 250
#ArchiveMaxCompressionRatio 300

# Use slower but memory efficient decompression algorithm.
# only affects the bzip2 decompressor.
# Default: disabled
#ArchiveLimitMemoryUsage

# Mark encrypted archives as viruses (Encrypted.Zip, Encrypted.RAR).
# Default: disabled
#ArchiveBlockEncrypted

# Mark archives as viruses (e.g. RAR.ExceededFileSize, Zip.ExceededFilesLimit)
# if ArchiveMaxFiles, ArchiveMaxFileSize, or ArchiveMaxRecursion limit is
# reached.
# Default: disabled
#ArchiveBlockMax


##
## Clamuko settings
## WARNING: This is experimental software. It is very likely it will hang
##                up your system!!!
##

# Enable Clamuko. Dazuko (/dev/dazuko) must be configured and running.
# Default: disabled
#ClamukoScanOnAccess

# Set access mask for Clamuko.
# Default: disabled
#ClamukoScanOnOpen
#ClamukoScanOnClose
#ClamukoScanOnExec

# Set the include paths (all files in them will be scanned). You can have
# multiple ClamukoIncludePath directives but each directory must be added
# in a seperate line.
# Default: disabled
#ClamukoIncludePath /home
#ClamukoIncludePath /students

# Set the exclude paths. All subdirectories are also excluded.
# Default: disabled
#ClamukoExcludePath /home/guru

# Don't scan files larger than ClamukoMaxFileSize
# Value of 0 disables the limit.
# Default: 5M
#ClamukoMaxFileSize 10M
```

APPENDICE C: local.cf

```
# SpamAssassin config file for version 3.x
# NOTE: NOT COMPATIBLE WITH VERSIONS 2.5 or 2.6
# See http://www.yrex.com/spam/spamconfig25.php for earlier versions
# Generated by http://www.yrex.com/spam/spamconfig.php (version 1.50)

# How many hits before a message is considered spam.
required_score          5.0

# Change the subject of suspected spam
rewrite_header subject          *****SPAM*****

# Encapsulate spam in an attachment (0=no, 1=yes, 2=safe)
report_safe             1

# Enable the Bayes system
use_bayes               1

# Enable Bayes auto-learning
bayes_auto_learn                1

# Enable or disable network checks
skip_rbl_checks         0
use_razor2              1
use_dcc                 1
use_pyzor               1

# Mail using languages used in these country codes will not be marked
# as being possibly spam in a foreign language.
ok_languages            all

# Mail using locales used in these country codes will not be marked
# as being possibly spam in a foreign language.
ok_locales              all
```

APPENDICE D: dovecod.conf

```
pop3_listen = [::]

# IP or host address where to listen in for SSL connections. Defaults
# to above non-SSL equilevants if not specified.
imaps_listen = 193.205.67.116:993
pop3s_listen = 193.205.67.116:995

# Disable SSL/TLS support.
#ssl_disable = no

# PEM encoded X.509 SSL/TLS certificate and private key. They're opened before
# dropping root privileges, so keep the key file unreadable by anyone but
# root. Included doc/mkcert.sh can be used to easily generate self-signed
# certificate, just make sure to update the domain
ssl_cert_file = /etc/pki/dovecot/dovecot.pem
ssl_key_file = /etc/pki/dovecot/private/dovecot.pem

# SSL parameter file. Master process generates this file for login processes.
# It contains Diffie Hellman and RSA parameters.
#ssl_parameters_file = /var/run/dovecot/ssl-parameters.dat

# How often to regenerate the SSL parameters file. Generation is quite CPU
# intensive operation. The value is in hours, 0 disables regeneration
# entirely.
#ssl_parameters_regenerate = 24

# Disable LOGIN command and all other plaintext authentications unless
# SSL/TLS is used (LOGINDISABLED capability). Note that 127.*.*.* and
# IPv6 ::1 addresses are considered secure, this setting has no effect if
# you connect from those addresses.
#disable_plaintext_auth = yes

# Use this logfile instead of syslog(). /dev/stderr can be used if you want to
# use stderr for logging (ONLY /dev/stderr - otherwise it is closed).
#log_path =

# For informational messages, use this logfile instead of the default
#info_log_path =

# Prefix for each line written to log file. % codes are in strftime(3)
# format.
#log_timestamp = "%b %d %H:%M:%S "

##
## Login processes
##

# Directory where authentication process places authentication UNIX sockets
# which login needs to be able to connect to. The sockets are created when
# running as root, so you don't have to worry about permissions. Note that
# everything in this directory is deleted when Dovecot is started.
login_dir = /var/run/dovecot-login

# chroot login process to the login_dir. Only reason not to do this is if you
# wish to run the whole Dovecot without roots.
# http://wiki.dovecot.org/Rootless
#login_chroot = yes


##
```

```
## IMAP login process
##

login = imap

# Executable location.
#login_executable = /usr/libexec/dovecot/imap-login

# User to use for the login process. Create a completely new user for this,
# and don't use it anywhere else. The user must also belong to a group where
# only it has access, it's used to control access for authentication process.
# Note that this user is NOT used to access mails.
# http://wiki.dovecot.org/UserIds
#login_user = dovecot

# Set max. process size in megabytes. If you don't use
# login_process_per_connection you might need to grow this.
#login_process_size = 32

# Should each login be processed in it's own process (yes), or should one
# login process be allowed to process multiple connections (no)? Yes is more
# secure, espcially with SSL/TLS enabled. No is faster since there's no need
# to create processes all the time.
#login_process_per_connection = yes

# Number of login processes to create. If login_process_per_user is
# yes, this is the number of extra processes waiting for users to log in.
#login_processes_count = 3

# Maximum number of extra login processes to create. The extra process count
# usually stays at login_processes_count, but when multiple users start logging
# in at the same time more extra processes are created. To prevent fork-bombing
# we check only once in a second if new processes should be created - if all
# of them are used at the time, we double their amount until limit set by this
# setting is reached. This setting is used only if login_process_per_use is yes.
#login_max_processes_count = 128

# Maximum number of connections allowed in login state. When this limit is
# reached, the oldest connections are dropped. If login_process_per_user
# is no, this is a per-process value, so the absolute maximum number of users
# logging in actually login_processes_count * max_logging_users.
#login_max_logging_users = 256

##
## POP3 login process
##

# Settings default to same as above, so you don't have to set anything
# unless you want to override them.

login = pop3

# Exception to above rule being the executable location.
#login_executable = /usr/libexec/dovecot/pop3-login

##
## Mail processes
##

# Maximum number of running mail processes. When this limit is reached,
# new users aren't allowed to log in.
#max_mail_processes = 1024

# Show more verbose process titles (in ps). Currently shows user name and
# IP address. Useful for seeing who are actually using the IMAP processes
# (eg. shared mailboxes or if same uid is used for multiple accounts).
```

```
#verbose_proctitle = no

# Show protocol level SSL errors.
#verbose_ssl = no

# Valid UID range for users, defaults to 500 and above. This is mostly
# to make sure that users can't log in as daemons or other system users.
# Note that denying root logins is hardcoded to dovecot binary and can't
# be done even if first_valid_uid is set to 0.
#first_valid_uid = 500
#last_valid_uid = 0

# Valid GID range for users, defaults to non-root/wheel. Users having
# non-valid GID as primary group ID aren't allowed to log in. If user
# belongs to supplementary groups with non-valid GIDs, those groups are
# not set.
#first_valid_gid = 1
#last_valid_gid = 0

# Grant access to these extra groups for mail processes. Typical use would be
# to give "mail" group write access to /var/mail to be able to create dotlocks.
#mail_extra_groups =

# ':' separated list of directories under which chrooting is allowed for mail
# processes (ie. /var/mail will allow chrooting to /var/mail/foo/bar too).
# This setting doesn't affect login_chroot or auth_chroot variables.
# WARNING: Never add directories here which local users can modify, that
# may lead to root exploit. Usually this should be done only if you don't
# allow shell access for users. See doc/configuration.txt for more information.
#valid_chroot_dirs =

# Default chroot directory for mail processes. This can be overridden by
# giving /./ in user's home directory (eg. /home/./user chroots into /home).
#mail_chroot =

# Default MAIL environment to use when it's not set. By leaving this empty
# dovecot tries to do some automatic detection as described in
# doc/mail-storages.txt. There's a few special variables you can use:
#
#   %u - username
#   %n - user part in user@domain, same as %u if there's no domain
#   %d - domain part in user@domain, empty if user there's no domain
#   %h - home directory
#
# You can also limit a width of string by giving the number of max. characters
# after the '%' character. For example %1u gives the first character of
# username. Some examples:
#
#   default_mail_env = maildir:/var/mail/%1u/%u/Maildir
#   default_mail_env = mbox:~/mail/:INBOX=/var/mail/%u
#   default_mail_env = mbox:/var/mail/%d/%n/:INDEX=/var/indexes/%d/%n
#
#default_mail_env =

# Space-separated list of fields to cache for all mails. Currently these
# fields are allowed followed by a list of commands they speed up:
#
#  Envelope      - FETCH ENVELOPE and SEARCH FROM, TO, CC, BCC, SUBJECT,
#                  SENTBEFORE, SENTON, SENTSINCE, HEADER MESSAGE-ID,
#                  HEADER IN-REPLY-TO
#  Body          - FETCH BODY
#  Bodystructure - FETCH BODY, BODYSTRUCTURE
#  MessagePart   - FETCH BODY[1.2.3] (ie. body parts), RFC822.SIZE,
#                  SEARCH SMALLER, LARGER, also speeds up BODY/BODYSTRUCTURE
#                  generation. This is always set with mbox mailboxes, and
#                  also default with Maildir.
```

```
#
# Different IMAP clients work in different ways, that's why Dovecot by default
# only caches MessagePart which speeds up most operations. Whenever client
# does something where caching could be used, the field is automatically marked
# to be cached later. For example after FETCH BODY the BODY will be cached
# for all new messages. Normally you should leave this alone, unless you know
# what most of your IMAP clients are. Caching more fields than needed makes
# the index files larger and generate useless I/O.
#
# With maildir there's one extra optimization - if nothing is cached, indexing
# the maildir becomes much faster since it's not opening any of the mail files.
# This could be useful if your IMAP clients access only new mails.

#mail_cache_fields = MessagePart

# Space-separated list of fields that Dovecot should never set to be cached.
# Useful if you want to save disk space at the cost of more I/O when the fields
# needed.
#mail_never_cache_fields =

# Workarounds for various client bugs:
#   oe6-fetch-no-newmail:
#     Never send EXISTS/RECENT when replying to FETCH command. Outlook Express
#     seems to think they are FETCH replies and gives user "Message no longer
#     in server" error. Note that OE6 still breaks even with this workaround
#     if synchronization is set to "Headers Only".
#   outlook-idle:
#     Outlook and Outlook Express never abort IDLE command, so if no mail
#     arrives in half a hour, Dovecot closes the connection. This is still
#     fine, except Outlook doesn't connect back so you don't see if new mail
#     arrives.
#   outlook-pop3-no-nuls:
#     Outlook and Outlook Express hang if mails contain NUL characters.
#     This setting replaces them with 0x80 character.
#client_workarounds =

# Dovecot can notify client of new mail in selected mailbox soon after it's
# received. This setting specifies the minimum interval in seconds between
# new mail notifications to client - internally they may be checked more or
# less often. Setting this to 0 disables the checking.
# NOTE: Evolution client breaks with this option when it's trying to APPEND.
#mailbox_check_interval = 0

# Like mailbox_check_interval, but used for IDLE command.
#mailbox_idle_check_interval = 30

# Allow full filesystem access to clients. There's no access checks other than
# what the operating system does for the active UID/GID. It works with both
# maildir and mboxes, allowing you to prefix mailboxes names with eg. /path/
# or ~user/.
#mail_full_filesystem_access = no

# Maximum allowed length for custom flag name. It's only forced when trying
# to create new flags.
#mail_max_flag_length = 50

# Save mails with CR+LF instead of plain LF. This makes sending those mails
# take less CPU, especially with sendfile() syscall with Linux and FreeBSD.
# But it also creates a bit more disk I/O which may just make it slower.
#mail_save_crlf = no

# Use mmap() instead of read() to read mail files. read() seems to be a bit
# faster with my Linux/x86 and it's better with NFS, so that's the default.
#mail_read_mmaped = no

# By default LIST command returns all entries in maildir beginning with dot.
```

```
# Enabling this option makes Dovecot return only entries which are directories.
# This is done by stat()ing each entry, so it causes more disk I/O.
# (For systems setting struct dirent->d_type, this check is free and it's
# done always regardless of this setting)
#maildir_stat_dirs = no

# Copy mail to another folders using hard links. This is much faster than
# actually copying the file. This is problematic only if something modifies
# the mail in one folder but doesn't want it modified in the others. I don't
# know any MUA which would modify mail files directly. IMAP protocol also
# requires that the mails don't change, so it would be problematic in any case.
# If you care about performance, enable it.
#maildir_copy_with_hardlinks = no

# Check if mails' content has been changed by external programs. This slows
# down things as extra stat() needs to be called for each file. If changes are
# noticed, the message is treated as a new message, since IMAP protocol
# specifies that existing messages are immutable.
#maildir_check_content_changes = no

# Which locking methods to use for locking mbox. There's three available:
#  dotlock: Create <mailbox>.lock file. This is the oldest and most NFS-safe
#           solution. If you want to use /var/mail/ like directory, the users
#           will need write access to that directory.
#  fcntl  : Use this if possible. Works with NFS too if lockd is used.
#  flock  : May not exist in all systems. Doesn't work with NFS.
#
# You can use both fcntl and flock too; if you do the order they're declared
# with is important to avoid deadlocks if other MTAs/MUAs are using both fcntl
# and flock. Some operating systems don't allow using both of them
# simultaneously, eg. BSDs. If dotlock is used, it's always created first.
mbox_locks = fcntl

# Should we create dotlock file even when we want only a read-lock? Setting
# this to yes hurts the performance when the mailbox is accessed simultaneously
# by multiple processes, but it's needed for reliable reading if no other
# locking methods are available.
#mbox_read_dotlock = no

# Maximum time in seconds to wait for lock (all of them) before aborting.
#mbox_lock_timeout = 300

# If dotlock exists but the mailbox isn't modified in any way, override the
# lock file after this many seconds.
#mbox_dotlock_change_timeout = 30

# umask to use for mail files and directories
#umask = 0077

# Drop all privileges before exec()ing the mail process. This is mostly
# meant for debugging, otherwise you don't get core dumps. Note that setting
# this to yes means that log file is opened as the logged in user, which
# might not work. It could also be a small security risk if you use single UID
# for multiple users, as the users could ptrace() each others processes then.
#mail_drop_priv_before_exec = no

##
## IMAP process
##

# Executable location
#imap_executable = /usr/libexec/dovecot/imap

# Set max. process size in megabytes. Most of the memory goes to mmap()ing
# files, so it shouldn't harm much even if this limit is set pretty high.
#imap_process_size = 256
```

```
# Support for dynamically loadable modules.
#imap_use_modules = no
#imap_modules = /usr/lib/dovecot/imap

##
## POP3 process
##

# Executable location
#pop3_executable = /usr/libexec/dovecot/pop3

# Set max. process size in megabytes. Most of the memory goes to mmap()ing
# files, so it shouldn't harm much even if this limit is set pretty high.
#pop3_process_size = 256

# Support for dynamically loadable modules.
#pop3_use_modules = no
#pop3_modules = /usr/lib/dovecot/pop3

##
## Authentication processes
##

# An Authentication process is a child process used by Dovecot that
# handles the authentication steps. The steps cover an authentication
# mechanism (auth_mechanisms, how the client authenticates in the IMAP or
# POP3 protocol), which password database should be queried (auth_passdb),
# and which user database should be queried (auth_userdb, to obtain
# UID, GID, and location of the user's mailbox/home directory).
#
# You can have multiple processes, though a typical configuration will
# have only one. Each time "auth = xx" is seen, a new process
# definition is started. The point of multiple processes is to be able
# to set stricter permissions. (See auth_user below.)
#
# Just remember that only one Authentication process is asked for the
# password, so you can't have different passwords accessible through
# different process definitions (unless they have different
# auth_mechanisms, and you're ok with having different password for
# each mechanisms).

# Authentication process name.
auth = default

# Specifies how the client authenticates in the IMAP protocol.
# Space separated list of permitted authentication mechanisms:
#   anonymous plain digest-md5 cram-md5
#
# anonymous - No authentication required.
# plain - The password is sent as plain text. All IMAP/POP3 clients
#  support this, and the password can be encrypted by Dovecot to match
#  any of the encryption schemes used in password databases.
# digest-md5 and cram-md5 - both encrypt the password so it is more
#  secure in transit, but are not well supported by clients, and
#  require that the password database use a matching encryption
#  scheme (or be in plaintext).
#
# See auth.txt for more details.
#
# If you are using SSL there is less benefit to digest-md5 and
# cram-md5 as the communication is already encrypted.
auth_mechanisms = plain

# Space separated list of realms for SASL authentication mechanisms that need
# them. You can leave it empty if you don't want to support multiple realms.
```

```
# Many clients simply use the first one listed here, so keep the default realm
# first.
#auth_realms =
# Default realm/domain to use if none was specified. This is used for both
# SASL realms and appending @domain to username in plaintext logins.
#auth_default_realm =
# User database specifies where mails are located and what user/group IDs
# own them. For single-UID configuration use "static".
# http://wiki.dovecot.org/Authentication
# http://wiki.dovecot.org/VirtualUsers
#   passwd: /etc/passwd or similiar, using getpwnam()
#   passwd-file <path>: passwd-like file with specified location
#   static uid=<uid> gid=<gid> home=<dir template>: static settings
#   vpopmail: vpopmail library
#   ldap <config path>: LDAP, see doc/dovecot-ldap.conf
#   pgsql <config path>: a PostgreSQL database, see doc/dovecot-pgsql.conf
auth_userdb = passwd
# Password database specifies only the passwords for users.
# http://wiki.dovecot.org/Authentication
#   passwd: /etc/passwd or similiar, using getpwnam()
#   shadow: /etc/shadow or similiar, using getspnam()
#   pam [<service> | *]: PAM authentication
#   passwd-file <path>: passwd-like file with specified location
#   vpopmail: vpopmail authentication
#   ldap <config path>: LDAP, see doc/dovecot-ldap.conf
#   pgsql <config path>: a PostgreSQL database, see doc/dovecot-pgsql.conf
auth_passdb = pam
#auth_executable = /usr/libexec/dovecot/dovecot-auth
# Set max. process size in megabytes.
#auth_process_size = 256
# User to use for the process. This user needs access to only user and
# password databases, nothing else. Only shadow and pam authentication
# requires roots, so use something else if possible. Note that passwd
# authentication with BSDs internally accesses shadow files, which also
# requires roots. Note that this user is NOT used to access mails.
# That user is specified by auth_userdb above.
auth_user = root
# Directory where to chroot the process. Most authentication backends don't
# work if this is set, and there's no point chrooting if auth_user is root.
#auth_chroot =
# Number of authentication processes to create
#auth_count = 1
# List of allowed characters in username. If the user-given username contains
# a character not listed in here, the login automatically fails. This is just
# an extra check to make sure user can't exploit any potential quote escaping
# vulnerabilities with SQL/LDAP databases. If you want to allow all characters,
# set this value to empty.
#auth_username_chars = abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ01234567890.-_@
# Username to use for users logging in with ANONYMOUS SASL mechanism
#auth_anonymous_username = anonymous
# More verbose logging. Useful for figuring out why authentication isn't
# working.
#auth_verbose = no
# Even more verbose logging for debugging purposes. Shows for example SQL
# queries.
#auth_debug = no
# digest-md5 authentication process. It requires special MD5 passwords which
# /etc/shadow and PAM doesn't support, so we never need roots to handle it.
# Note that the passwd-file is opened before chrooting and dropping roots in dovecot-openssl.cnf
# privileges, so it may be 0600-root owned file.
#auth = digest_md5
#auth_mechanisms = digest-md5
#auth_realms =
#auth_userdb = passwd-file /etc/passwd.imap
#auth_passdb = passwd-file /etc/passwd.imap
#auth_user = imapauth
```

```
#auth_chroot =
# if you plan to use only passwd-file, you don't need the two auth processes,
# simply set "auth_methods = plain digest-md5"
```

APPENDICE E: utente

```
#!/bin/sh
echo login:$1 UID:$2 $2x$1
/usr/sbin/useradd -m -d /home/users/$1 -g 100  -p "$2x$1" -s /bin/rbash -u $2 -l $1
chmod 700 /home/users/$1
echo $1@zeus.to.astro.it > /home/users/$1/.forward
chown $1:100  /home/users/$1/.forward
echo "=============================="
cat /etc/passwd |grep $1
ls /home/users |grep $1
echo "=============================="
exit
```

APPENDICE F: sposta_la_posta

```
#!/bin/sh
# Check that an argument was supply
if [ $# -eq 0 ]
then
echo "usage : sposta_la_posta user"
exit
fi
# Check that Zeus mounts correctly Sam directories
if [ -f /migrazione/home/.OK ]
then
echo "directory migrazione/home correctly mount"
else
echo "please mount directory migrazione/home" && exit
fi
if [ -f /migrazione/spool/.OK ]
then
echo "directory migrazione/spool correctly mount"
else
echo "please mount directory migrazione/spool"  && exit
fi
cp -pr /home/users/$1 /migrazione/home/.
cp -pr /var/spool/mail/$1 /migrazione/spool/.
echo $1@sam.oato.inaf.it > /home/users/$1/.forward
chmod 600 /home/users/$1/.forward
chown $1 /home/users/$1/.forward
#usermod -L $1
echo "OK"
```

LISTA DEGLI ACRONIMI

BSD Berkeley Software Distribution
IMAP Internet Message Access Protocol oppure Interactive Mail Access Protocol
LDA Local delivery Agent
MTA Mail Transport Agent
MUA Mail User Agent
NFS Network File System
POP Post Office Protocol
SMTP Simple Mail Transfer Protocol
SSH Secure SHell
SSL Secure Sockets Layer
TLS Transport Layer Security