Majordomo: How I Manage 17 Mailing Lists Without Answering "-request" Mail

D. Brent Chapman – Great Circle Associates

ABSTRACT

Majordomo is a perl program written to handle routine administration of Internet mailing lists with as little human intervention as possible. Modeled after the Listserv implementations common on BITNET (but unfortunately rare on the Internet), it automates the administration of mailing lists by allowing users to perform the most frequent operations ("subscribe" and "unsubscribe") themselves, while allowing the list owners to either "approve" each of these operations (or initiate them on behalf of a user), or merely monitor them as they are automatically approved. It also automates response to certain other common queries from users, such as "what lists are served by this Majordomo server?", "what is the topic of list 'foobar'?", "who is already on list 'foobar'?", and "which lists managed by this Majordomo server am I already on?".

Majordomo allows individual list owners to manage their own lists (subscribe and unsubscribe users, and change the general information message for their list) without any action by the overall Majordomo owner. It serves both "open" lists (where users can add themselves to the list, and the list owner is merely informed of this action) and "closed" lists (where a subscription request from a user generates an approval request from the Majordomo server to the list owner, who can then either approve or ignore the request).

Finally, all interactions with Majordomo by both users and list owners take place totally by electronic mail, so users and list owners do not require login access (nor even direct TCP/IP connectivity) to the machine Majordomo is running on, and no special client software is required.

Introduction

Anyone who has ever managed a significant electronic mailing list by hand (which is, on the Internet at least, the usual method) knows how much time it takes to process the endless requests from users of the form "please subscribe me to your list", "please unsubscribe me from your list", "please tell me about your list", "please tell me if I'm already on your list", and so forth. It's a time-consuming, boring, repetitive task; just the sort of thing that's a perfect candidate to be automated.

When SAGE (the System Administrators Guild, a USENIX Special Technical Group) was formed, the founding members decided to establish over a dozen mailing lists for various purposes (one for the board of directors, one for each of the 16 initial working groups, one the chairs of all the working groups, and so forth). The USENIX Association volunteered the USENIX.ORG machine as a home for these mailing lists, but didn't have the staff resources to set up and operate the mailing lists. I volunteered to act as Postmaster for SAGE, and handle all the mailing lists. As an independent consultant, my schedule is rather erratic, and I don't have a company paying my salary while I pursue volunteer work like this; thus, I wished to automate the job as much as possible, so that I could provide a high level of service to the users (including fast turnaround on their requests) while spending as little time as possible in the long run on administrivia. A BITNET-style Listserv seemed to be an appropriate solution, so I started investigating alternatives.

Defining the Problem

The first step was to identify just what functionality I desired. First and foremost, I wanted something that would handle routine "subscribe" and "unsubscribe" requests automatically, with no human intervention required for routine requests (though I wanted to give the owner of a given list the option of passing judgement on all subscription requests, if they so desired). Second, I wanted something that could easily handle many mailing lists simultaneously; I had 17 to begin with, and I was sure that more would be added as time passed. Third, I wanted something that could automatically handle other user requests (such as "what lists are available?", "please tell me about list 'foobar'", and "which of your lists am I on?") that, while less common than "subscribe" and "unsubscribe", still occur relatively frequently.

The first thing I did was look around for suitable publicly available software that might already exist, or that might be easily adapted to my needs. Searches of the common Internet software archives, queries to the "Archie" anonymous FTP indexing service, and email to certain acquaintances who I thought might know of such software produced two results: an implementation of the BITNET Listserv written in C for UNIX (from the comp.sources.unix archives), and several different programs named "listserv" written in perl.

I first examined the BITNET Listserv C package from the comp.sources.unix newsgroup archives. It looked like it would do most of what I wanted, but it also looked like it did a lot of things I didn't really care about (there appeared to be features for coordinating activities between multiple Listserv servers on different machines, for instance). It appeared to be rather short on documentation, and what documentation there was seemed to assume that the reader was already familiar with BITNET Listserv implementation and operation. All in all, it looked like it would be a real headache for me to install, configure, and maintain, since I'm *not* familiar with BITNET Listserv implementation and operation.

The next things I looked at were several perl scripts from a variety of sources that were supposedly Listserv-like servers. Some of these scripts were pointed out to me by folks on the net who knew I was looking for such a thing, and I found others by searching through Archie for "listserv". Unfortunately, these various scripts all turned out to be more what I'd call "archive servers" than "listserv" implementations; they were written to automate retrieval of files from archives via email, for folks who don't have access to anonymous FTP. When I examined one of these scripts that claimed to support "subscribe" and "unsubscribe" requests, I found that what it did with such requests was forward them by email to the mailing list owner for manual processing; this was exactly what I was trying to avoid!

In the end, I decided to implement my own version of Listserv, so that I could get exactly what I wanted. The name for my software was provided by Eliot Lear of Silicon Graphics, Inc.; he suggested "majordomo", which the dictionary defines as "a person who speaks, makes arrangements, or takes charge for another", and which seems perfectly appropriate given the nature of the software.

Designing a Solution

My first step in designing a solution was to decide on the general approach I was going to take. First, I decided that all routine interactions with Majordomo would take place asynchronously via email. Second, since the software was going to spend most of its time parsing emailed instructions, processing text files (the actual mailing lists) according to those instructions, and generating emailed responses to users, I wanted to write it in a language well-suited for that task; perl seemed the natural choice.

In the Majordomo world model, there are three types of people: users (without any special privileges), mailing list owners, and the owner of the Majordomo server itself. Interactions with users take place strictly by email; the user mails a set of requests to Majordomo, and Majordomo processes those requests and sends back appropriate replies. Interactions with list owners also take place strictly by email, but a list owner can do a few things that a normal user can't; the commands that are restricted to list owners are protected with a per-list password (though it's very weak password protection, since the password is passed in the clear through the email; the goal is not absolute security, but to avoid people making a nuisance of themselves by abusing the Majordomo server). The Majordomo owner is the person responsible for maintaining the Majordomo server itself, and for performing tasks such as creating new mailing lists to be served by Majordomo.

The software needs to support multiple mailing lists, each owned by different individuals. Some owners wish to approve all "subscribe" requests for their list (a "closed" list), while other owners wish routine "subscribe" requests to be approved automatically (an "open" list), with notification to the owner.

Command	**Description**
subscribe *list* [*address*]	Subscribe yourself (or *address*, if specified) to *list*
unsubscribe *list* [*address*]	Unsubscribe yourself (or *address*, if specified) from *list*
which [*address*]	Find out which lists you (or *address*, if specified) are on
who *list*	Show the members of *list*
info *list*	Show the general introductory information for *list*
lists	Show the lists handled by this Majordomo server
help	Retrieve a help message, explaining these commands
end	Stop processing commands (useful if your mailer automatically adds a signature to your messages)

Figure 1: Majordomo user commands

Routine "unsubscribe" requests are approved automatically, with notification to the list owner, for both open and closed lists. Owners have a way (the "approve" command) to approve all "subscribe" requests on closed lists, as well as non-routine "subscribe" and "unsubscribe" requests on open lists. A "non-routine request" is one that affects a different address than the request appears to originate from; for instance, a request from "joe@foobar.com" to subscribe or unsubscribe "alice@foobar.com" is a non-routine request. All non-routine requests (on both open and closed lists) are forwarded to the list owner for approval.

Majordomo accepts the commands shown in Figure 1 from any user. In addition, Majordomo accepts the password-protected commands shown in Figure 2, which are for use by list owners to manage their list. Authentication is based solely on knowledge of the password for the list in question; no attempt is made to check that the address of the person issuing the command is the same as the address of the list owner. As mentioned earlier, the goal of the minimal security in Majordomo is to prevent anti-social people from making a nuisance of themselves; I don't make any claims that the security is particularly strong.

A side benefit of authentication by password is that the owner can manage their list from any of their accounts; they don't have to always use the same account on a certain machine, for instance. The "owner" of a given list could in fact be an alias for multiple people, any of whom could approve requests for the list. Because the owner of a list is always notified of successful "subscribe" and "unsubscribe" requests concerning their list, even if the owner initiated those requests on behalf of a user, multiple owners would automatically be kept up to date on each other's actions concerning the list.

Note that the "approve" command is simply "approve *password*" prepended to a "subscribe" or "unsubscribe" request. This simplifies command processing; in handling an "approve" message, the command processor checks that the password is correct for the list being acted on, then recursively processes the "subscribe" or "unsubscribe" command with a flag set that tells the processor that the operation is pre-approved and should simply be carried out, even if it is a non-routine request. The right way to think about "approve", by the way, is that the list owner is telling Majordomo "I approve this command; just do it!", not "I approve this request you sent me earlier". Majordomo doesn't keep track of outstanding requests; when an "approve" command comes in from a list owner, Majordomo doesn't check to see that the owner is approving something Majordomo had previously requested, or anything like that. A list owner can thus issue "approve" commands on behalf of a user (to drop a dead account from the list, for instance) without any prior action by the user.

An important distinction that many people misunderstand is the difference between managing a mailing list, and managing the traffic on a mailing list. Managing a mailing list (which is what Majordomo does) means exactly that: managing a list of names. Managing the traffic on a mailing list (which is commonly called "moderating" the mailing list) means either automatically or manually reviewing each message that is submitted for the list, then either forwarding it to the list (perhaps after header or content editing, depending on the nature of the mailing list) or discarding it. The changes made to messages before forwarding them to such a moderated mailing list can be as simple as rewriting the headers of the message to arrange for errors to come back to the list owner, or as complex as completely rewriting the body of the message to preserve the anonymity of the originator. Editorial policies (such as only forwarding messages to the list that were sent by a member of the list, and refusing messages from "outsiders") might also be enforced automatically or manually. All of this is outside the scope of Majordomo; all Majordomo does is maintain the file containing the list of email addresses. How that list is used (whether it is simply included as an alias in the `/etc/aliases` file, or used by a forwarding that enforces a "no messages from non-members policy" as described above, or whatever) is not something for Majordomo to determine.

Implementing the Proposal

Once I had more or less decided what I wanted to implement and how, I sat down to the nitty-gritty details of getting it done. It took about 2 days of concentrated work to write the core of the program, followed by a test installation and another couple of

Command	**Description**
approve *password* {subscribe | unsubscribe} *list address*	Approve a non-routine subscribe or unsubscribe request concerning *list*
newinfo *list password*	Provide a new "info" message for *list*, to be sent in response to "info" and "subscribe" requests
passwd *list old-password new-password*	Change the password for *list*

Figure 2: Majordomo list owner commands

days of on-again, off-again testing and enhancement. All told, I spent about 20 hours on the project, and ended up with about than 600 lines of perl code that implemented almost all the features listed above (I didn't implement "which" and "unsubscribe" until a couple of weeks later). This was the version that was initially installed on USENIX.ORG to run the SAGE mailing lists in late June, 1992. Over the next couple of weeks, I spent another 20 or so hours implementing the remaining commands, fixing minor bugs, and generally cleaning up the program. I've continued to make minor enhancements since then. Today, the program stands at 815 lines of perl code, not including libraries.

While writing Majordomo, I made extensive use of other people's work that had been previously released on the net, including software to process mail headers and perform file locking. From one of the perl archives on the Internet, I obtained a perl package called "mailstuff.pl" (written by Gene Spafford) which parses RFC822 mail headers into perl associative arrays for easy processing; with a few minor modifications, it was just what I needed to handle all the mail header processing for Majordomo.

I needed a safe way for Majordomo to lock files while editing them (adding or deleting users on a mailing list, or changing the "info" file for a list, for instance), to prevent multiple Majordomo processes from tripping each other up. I was familiar with Erik Fair's "shlock" program, which is provided in the NNTP distribution as a file locking mechanism for use in shell scripts, and knew it would provide the kind of locking I wanted; porting the code from a stand-alone C program to a 150-line perl package was a relatively simple matter. The biggest problem I encountered was that the C code used "goto" to break out of nested command logic when exceptions occurred; unlike some, I don't dogmaticly object to "goto" on general principles, but this particular usage of "goto" simply isn't supported in perl.

Other complications included addressing and appropriate case sensitivity. It was slightly tricky to get all the "To:" and "From:" addresses correct on mail generated by Majordomo, so that replies to commands and requests for approval from Majordomo went to the right place, and could themselves be replied to with appropriate results. It was also tricky to get certain things to be case sensitive (passwords, for example), and other things to be case insensitive (email addresses, mailing list names, and commands, for instance); further, some case insensitive items (such as mailing list names) need to be smashed to lower case before use, while others (such as email addresses) need to be preserved in mixed case and merely compared in a case insensitive manner.

Because it needs to edit files (the mailing lists, the "info" files for each list, and so forth), I decided that Majordomo needed to run setgid to a specially-created group which would have appropriate permissions on those files. Perl includes a nifty dataflow-tracing feature (commonly known as "taintperl") that is automatically activated when a perl script is run setuid or setgid; this feature attempts to ensure that the script doesn't do anything "dangerous". The perl on-line manual page describes this feature:

> When perl is executing a setuid script, it takes special precautions to prevent you from falling into any obvious traps. (In some ways, a perl script is more secure than the corresponding C program.) Any command line argument, environment variable, or input is marked as "tainted", and may not be used, directly or indirectly, in any command that invokes a subshell, or in any command that modifies files, directories or processes. Any variable that is set within an expression that has previously referenced a tainted value also becomes tainted (even if it is logically impossible for the tainted value to influence the variable).

While this is certainly a valuable feature of perl, I wasn't able to get Majordomo to function because of it. I spent many hours trying to make "taintperl" happy before I gave up and wrote a simple C "wrapper" program that sets the real UID and GID to the effective UID and GID before executing the Majordomo perl script, thus not activating the "taintperl" feature. This is almost certainly *not* the right thing to do; at some point, I need to go back and figure out how to make Majordomo work under "taintperl". Particularly since I'm bypassing the "taintperl" security features, Majordomo makes a special effort to validate user input (email addresses and mailing list names, for instance) and ensure that it doesn't contain anything dangerous (a command

```
$whereami = "GreatCircle.COM";
$whoami = "Majordomo@$whereami";
$whoami_owner = "Majordomo-Owner@$whereami";
$homedir = "/usr/local/majordomo";
$listdir = "$homedir/Lists";
$log = "$homedir/Log";
```

Figure 3: Sample `/etc/majordomo.cf` file

like "|uudecode" in an email address or an absolute path name like "/etc/passwd" as a mailing list name) before using that input to interact with the operating system (by opening files by that name, and so forth).

The title of this paper states that I don't answer "-request" mail (that is, mail people send to "*list*-request" with requests concerning *list*). While that's true, *something* has to answer "-request" mail. Mail sent to "*list*-request" can't simply be forwarded to Majordomo for processing, since it almost certainly doesn't contain commands that Majordomo would understand. A simple little perl script called "request-recording" (abbreviated as "request-rec" in Figure 4) answers the "-request" mail for each mailing list, and sends back a message (customized to the list in question) telling the user how to use Majordomo to subscribe to the list, get information about the list, or get a copy of Majordomo's help file; in addition, instructions are provided on how to reach a human being, just in case.

Configuring Majordomo

At startup, Majordomo reads a configuration file (as specified by the "MAJORDOMO_CF" environment variable or on the command line, or "/etc/majordomo.cf" by default) that provides site-specific information, including the name of the site, who mail from Majordomo should appear to be from, where Majordomo's supporting programs are located, where the lists Majordomo manages are located, and where Majordomo's log is located. Figure 3 shows a sample Majordomo configuration file. All Majordomo-managed files (the lists themselves, and the "info" and "password" information for those lists) are kept in a directory specified by the "$listdir" variable in the configuration file. Each mailing list is kept in a file in the $listdir directory that is exactly the name of the mailing list. Mailing list names may contain only lower case letters, numbers, "-", and "_". The lists Majordomo thinks it manages are the files in $listdir whose names meet these criteria for mailing list names. There is no specific "list of lists" in a file anywhere; thus, creating a new list for Majordomo to manage merely involves creating a new file with appropriate permissions in $listdir and creating appropriate entries in either /etc/aliases or /usr/lib/aliases to use that file.

Several auxiliary files may be associated with each list in $listdir. The password for *list* is contained in the file "*list*.passwd". The descriptive info for *list* (which will be returned in response to a "info *list*" or "subscribe *list*" command) is in "*list*.info". The existence of a file called "*list*.closed" indicates that *list* is a "closed" list, and that all "subscribe *list*" requests must be approved by the list owner. Note that the names of these auxiliary files are invalid mailing list names, because they contain a "."; that's how Majordomo differentiates the mailing list files from the auxiliary files.

Majordomo is closely tied to the /etc/aliases or /usr/lib/aliases file. A number of aliases are required for the Majordomo server itself, as well as for each of the lists managed by Majordomo. Figure 4 shows sample entries for the /etc/aliases file on a machine using Majordomo to run two lists ("open-list" and "closed-list"). The "-approval" alias is where Majordomo will send requests for approval for actions concerning a list. The "owner-" alias is not used by Majordomo, but is used by Sendmail to notify the owner of a mailing list of problems with that mailing list (bounced messages, and so forth; see the Sendmail documentation for more information). The "owner-" and "-approval" aliases could point to different people; each could also expand to multiple people.

Using Majordomo

To use Majordomo, a user sends commands as an email message to the address the Majordomo server is configured to recognize (for the sample configuration in Figure 3, the address is "Majordomo@GreatCircle.COM"). For instance, to find out what lists are served by Majordomo@GreatCircle.COM, a user named "Jane@Somewhere.ORG" might send the following

```
majordomo: "|/usr/local/majordomo/wrapper /usr/local/majordomo/majordomo"
owner-majordomo: brent

open-list: :include:/usr/local/majordomo/Lists/open-list
open-list-request: "|/usr/local/majordomo/wrapper /usr/local/majordomo/request-rec open-list"
open-list-approval: joe@foobar.com
owner-open-list: joe@foobar.com

closed-list: :include:/usr/local/majordomo/Lists/closed-list
closed-list-request: "|/usr/local/majordomo/wrapper /usr/local/majordomo/request-rec closed-list"
closed-list-approval: bob@elsewhere.edu
owner-closed-list: bob@elsewhere.edu
```

Figure 4: Sample /etc/aliases entries

message:

```
From: Jane@Somewhere.ORG
To: Majordomo@GreatCircle.COM

lists
```

The "Subject:" line of a message, if any, is ignored by Majordomo, so there’s no harm in leaving it out. Jane would receive a message like this in response to her query:

```
From: Majordomo@GreatCircle.COM
To: Jane@Somewhere.ORG
Subject: Majordomo results

>>>> lists
Majordomo@GreatCircle.COM serves the
following lists:

    majordomo-announce
    majordomo-users

Use the ’info <list>’ command to get
more information about a specific list.
```

Upon receiving this, Jane might wish to find out more about each of these lists. She could send the following request:

```
From: Jane@Somewhere.ORG
To: Majordomo@GreatCircle.COM

info majordomo-announce
info majordomo-users
```

In return, Majordomo would respond with:

```
From: Majordomo@GreatCircle.COM
To: Jane@Somewhere.ORG
Subject: Majordomo results

>>>> info majordomo-users
This list is for discussions (including
bug reports, enhancement reports,
and general usage tips) concerning
the Majordomo mailing list manager.
...

>>>> info majordomo-announce
This list is for announcements of new
releases of the Majordomo mailing
list manager.
...
```

If Jane wishes to subscribe to one of the lists (say, the majordomo-users list), she would send the following request:

```
From: Jane@Somewhere.ORG
To: Majordomo@GreatCircle.COM

subscribe majordomo-users
```

In return, she would receive two messages. The first is a standard Majordomo response:

```
From: Majordomo@GreatCircle.COM
To: Jane@Somewhere.ORG
Subject: Majordomo results

>>>> subscribe majordomo-users
Succeeded.
```

The second is "welcome" message with specific information concerning the list (note that it also includes the same information that an "info" command on the list would return). This message goes to the subscribed address, not the address the request was made from (though in this case those are the same; since Jane didn’t specify an address to subscribe, it defaulted to the address the request was made from):

```
From: Majordomo@GreatCircle.COM
To: Jane@Somewhere.ORG
Subject: Welcome to majordomo-users

Welcome to the majordomo-users mailing list!

If you ever want to remove yourself
from this mailing list, send the
following command in email to
"Majordomo@GreatCircle.COM":

    unsubscribe majordomo-users \
       Jane@Somewhere.ORG

Here’s the general information for the
list you’ve subscribed to, in case you
don’t already have it:

This list is for discussions (including
bug reports, enhancement reports,
and general usage tips) concerning
the Majordomo mailing list manager.
...
```

At the same time, the owner of the list (through the "majordomo-users-approval" alias in the `/etc/aliases` file on the Majordomo machine) would receive the following notification of a new user:

```
From: Majordomo@GreatCircle.COM
To: majordomo-users-approval@GreatCircle.COM
Subject: SUBSCRIBE majordomo-users

Jane@Somewhere.ORG has been
added to majordomo-users.
No action is required on your part.
```

If Jane wanted to subscribe some other address to majordomo-announce (the email address "SysStaff@Somewhere.ORG", for instance, so that all members of the system staff would receive announcements concerning Majordomo), she could submit the following request:

```
From: Jane@Somewhere.ORG
To: Majordomo@GreatCircle.COM

subscribe majordomo-announce \
       SysStaff@Somewhere.ORG
```

This would cause the following message to be returned to Jane:

```
From: Majordomo@GreatCircle.COM
To: Jane@Somewhere.ORG
Subject: Majordomo results

>>>> subscribe majordomo-announce \
       SysStaff@Somewhere.ORG
Your request to Majordomo@GreatCircle.COM:

    subscribe majordomo-announce \
       SysStaff@Somewhere.ORG

has been forwarded to the owner of the
"majordomo-announce" list for approval.
This could be for any of several reasons:

    You might have asked to subscribe to a
    "closed" list, where all new additions
```

```
        must be approved by the list owner.

        You might have asked to subscribe or
        unsubscribe an address other than
        the one that appears in the headers
        of your mail message.

    When the list owner approves your request, you
    will be notified.

    If you have any questions about the
    policy of the list owner, please contact
    "majordomo-announce-approval@GreatCircle.COM".
```

At the same time, Majordomo sends the following message to the mailing list owner:

```
From: Majordomo@GreatCircle.COM
To: majordomo-announce-approval@GreatCircle.COM
Subject: APPROVE majordomo-announce

Jane@Somewhere.ORG requests that you
approve the following:

    subscribe majordomo-announce \
        SysStaff@Somewhere.ORG

If you approve, please send a message
such as the following back to
Majordomo@GreatCircle.COM (with the
appropriate PASSWORD filled in,
of course):

    approve PASSWORD subscribe \
        majordomo-announce SysStaff@Somewhere.ORG

If you disapprove, do nothing.
```

If the list owner sends such an "approve" command back to Majordomo, and the password is the correct password for the list in question, then the addition will take place. The address being subscribed (SysStaff@Somewhere.ORG, in this case) will receive a standard "Welcome to majordomo-announce" message and the list owner will receive a standard "SUBSCRIBE" notification, as shown above.

Such an "approve" cycle takes place if a user attempts to subscribe or unsubscribe any address that doesn't match the one in the header of their message, or if a user asks to subscribe to a "closed" list.

To find out who is on the majordomo-users list, Jane would send the following request:

```
From: Jane@Somewhere.ORG
To: Majordomo@GreatCircle.COM

who majordomo-users
```

and would receive the following response:

```
From: Majordomo@GreatCircle.COM
To: Jane@Somewhere.ORG
Subject: Majordomo results

>>>> who majordomo-users
Members of list 'majordomo-users':

brent@GreatCircle.COM (Brent Chapman)
Jane@Somewhere.ORG
Joe User <Joe@Elsewhere.GOV>
...
```

To find out which of the lists she's on that are served by a given Majordomo server, Jane would send the following request:

```
From: Jane@Somewhere.ORG
To: Majordomo@GreatCircle.COM

which
```

Majordomo would respond with:

```
From: Majordomo@GreatCircle.COM
To: Jane@Somewhere.ORG
Subject: Majordomo results

>>>> which
The address 'Jane@Somewhere.ORG' is
on the following lists served by
Majordomo@GreatCircle.COM:

        majordomo-users
```

To unsubscribe herself from the majordomo-users list, Jane would send a request such as:

```
From: Jane@Somewhere.ORG
To: Majordomo@GreatCircle.COM

unsubscribe majordomo-users \
        Jane@Somewhere.ORG
```

To which Majordomo would respond:

```
From: Majordomo@GreatCircle.COM
To: Jane@Somewhere.ORG
Subject: Majordomo results

>>>> unsubscribe majordomo-users \
        Jane@Somewhere.ORG
Succeeded.
```

The following message would also be sent to the list owner:

```
From: Majordomo@GreatCircle.COM
To: majordomo-users-approval@GreatCircle.COM
Subject: UNSUBSCRIBE majordomo-users

Jane@Somewhere.ORG has unsubscribed
from majordomo-users.
No action is required on your part.
```

If Jane's mailer automatically appended a signature to the end of all her outgoing messages, she could issue the "end" command as the last command of her messages to cause Majordomo to stop processing at that point. In addition, she could include blank lines or comments (anything following a '#' on a line is a comment, and is discarded before the line is processed) if she wanted to.

If the owner of the "majordomo-users" list wished to change the information file that is sent in response to "info" and "subscribe" requests, he could do that with a message such as:

```
To: Majordomo@GreatCircle.COM

newinfo majordomo-users PASSWORD
This is a revised information file
for the majordomo-users mailing list.
END
```

If the password used was the correct password for the list, Majordomo would replace the existing info file with the contents of the message to the "END" marker (or the end of the message, if there was no marker). A wise list owner would probably include an "info majordomo-users" command after the "END" marker so that he could verify that the information update succeeded.

A list owner could also use a message like this to change the password for their list:

```
To: Majordomo@GreatCircle.COM
passwd majordomo-users OLD NEW
```

If the old password for majordomo-users was "OLD", then Majordomo would change the password to "NEW". For all Majordomo list owner operations that require passwords, knowledge of the password for the list is the sole authentication performed on the command. As I've said elsewhere in this paper, this isn't intended to be highly secure; it's merely intended to keep obnoxious people from making a nuisance of themselves by abusing list owner commands.

Note that Majordomo does not yet support continuation lines (a command line that ends with a backslash, indicating that the command continues on the next line) as shown above, though it is high on the list of features to be added. Continuation lines were used here for typesetting reasons.

Experiences with Majordomo

Majordomo is currently used to run the 17 SAGE mailing lists on USENIX.ORG, and to run the "Majordomo-Users" and "Majordomo-Announce" mailing lists at GreatCircle.COM (see the "Availability" section for more information about these lists). It's been in operation on USENIX.ORG since late June, 1992. In the two months between then and the time this paper was written, it has processed almost 1800 requests, all without encountering any major bugs or problems (though a number of minor bugs have been found and corrected). A number of other sites requested and received beta-test versions of the program, but I haven't heard back from any of them that they've begun using the software yet.

While Majordomo is similar to and inspired by Listserv, I haven't really attempted to make it a Listserv clone. I've chosen to use many of the same commands as Listserv, but I've often used slightly different syntaxes for some commands; for instance, the Listserv syntax for "subscribe" is "subscribe *list real_name*", as opposed to the Majordomo syntax of "subscribe *list* [*address*]". This may not have been a good idea; perhaps I should have either made the Majordomo syntax identical to the Listserv syntax or made it completely different. The copy of Majordomo running on USENIX.ORG uses the email address "Listserv", not "Majordomo"; it's not clear if that was a good idea, since it's not really Listserv.

Future Work

The next major set of features I intend to add are to support email retrieval of files through Majordomo. I need to look at mechanisms and syntaxes for making files and directories readable, writable, and searchable via email. I intend to support the notion of "open" and "closed" file directories (similar to the "open" and "closed" mailing lists currently implemented); only authorized people (where authorization might be determined by knowledge of an appropriate password, or by membership on a mailing list associated with the directory) will be able to retrieve files from "closed" directories. I also intend to support "writable" and "read-only" directories and files. I'm going to consider special support specificly for mailing list archives, to allow users to request only messages matching certain patterns or containing specified keywords from a given archive, rather than forcing them to retrieve the whole archive and do the search themselves.

At some point, I (or someone else) should go back in and make Majordomo work under "taintperl", so that the "wrapper" program won't be necessary. I firmly believe that "taintperl" is good and valuable, and that operating under it would improve the security of Majordomo; I just didn't have the time to work out all the details during my initial implementation phase.

I'd like to add a number of minor features to the program, including suppression of duplicate addresses in mailing lists (but is "joe@foobar.com" the same as "joe@workstation.foobar.com"?), recognition of unambiguous command abbreviations, support for continuation lines (some mailers insist on auto-wrapping text to fit an 80-column display; while this is often preferable to paragraph-long lines in text messages, it wreaks havoc with long Majordomo commands), support for a command indicating what return address Majordomo should use for its replies (for use by folks whose mailers generate broken reply addresses in the headers; this might, however, have security implications that would need to be carefully considered), and support for commands in the "Subject:" line of the message. I might look at making Majordomo more Listserv-compatible.

Availability

The package is available for anonymous FTP on machine FTP.GreatCircle.COM, in file "pub/majordomo.tar.Z". If you do not have anonymous FTP access, contact me (contact information is in the "Author Information" section, below), and I'll try to get a copy to you by email or some other means.

If you install Majordomo, please add yourself to the mailing list Majordomo-Users@GreatCircle.COM, which is for discussions concerning use of, problems with, and enhancements for Majordomo. Announcements of new releases of Majordomo will be sent to Majordomo-Announce@GreatCircle.COM. You can add yourself to either or both lists by sending appropriate Majordomo commands to the electronic mail alias Majordomo@GreatCircle.COM.

Author Information

Brent Chapman is a consultant in the San Francisco Bay Area, specializing in the configuration, operation, and networking of UNIX systems. He is also currently Postmaster for SAGE (the USENIX Special Technical Group focusing on system administration issues). During the last several years, he has been an operations manager for a financial services company, a world-renowned corporate research lab, a software engineering company, and a hardware engineering company. He holds a Bachelor of Science degree in Electrical Engineering and Computer Science from the University of California, Berkeley. He can be contacted by electronic mail to Brent@GreatCircle.COM, by phone at +1 415 962 0841, by FAX at +1 415 962 0842, or by U.S. Mail to Great Circle Associates, 1057 West Dana St., Mountain View, CA 94041.