processor

Version 8.0

July 16, 1996

CREN
Corporation for Research and Educational Networking
1112 16th Street NW; Suite 600
Washington, D.C. 20036
202-872-4200

Table of Contents

1. Preface

1.1. Purpose of This Document

The purpose of this document is to explain the various issues involved in installing and managing a ListProc 8.0 site. It does *not* include instructions for list subscribers or list managers. CREN provides other manuals for these purposes.

1.2. ListProc Documentation

CREN provides five documents for ListProc. All are available from CREN's anonymous ftp site (ftp.cren.net) in the /listproc directory. They may also be found on CREN's World Wide Web and gopher servers. (http://www.cren.net and gopher.cren.net, respectively.) Each document is available in both postscript and Rich Text formats. The documents, and their corresponding file names are listed below:

User Manual (userman.ps or userman.rtf)
A comprehensive description of how to subscribe and change personal settings for ListProc lists.

User Reference Card (usercard.ps or usercard.rtf)
A brief summary of ListProc's user commands.

List Owner Manual (ownerman.ps or ownerman.rtf)
Complete instructions for the various aspects of managing a ListProc list.

List Owner Reference Card (ownercard.ps or ownercard.rtf)
A summary of ListProc's owner commands.

Site Reference (siteref.ps or siteref.rtf)
Technical instructions for maintaining a ListProc installation. (This document.)

1.3. Obtaining Technical Support

Technical support for ListProc 8.0 site managers is primarily through the CREN-LISTPROC mailing list. To subscribe, send e-mail to ListProc@list.cren.net with the following one line message:

```
sub cren-listproc your-first-name your-last-name
```

Archives of the list are available via anonymous ftp from ftp.cren.net in the /archives/cren-listproc directory.

Please direct problems, bugs and questions to this mailing list so everyone can benefit from your experience. Unless specifically requested not to, we will summarize any

private e-mail and/or telephone conversations and post them to the mailing list. Since ListProc runs on different systems, please indicate in your questions and comments the hardware platform, hardware configuration (amount of RAM/ hard disk), operating system and operating system release you are using. Also include any relevant error messages and /or log entries.

2. ListProc Overview

CREN's ListProc is an automated information distribution and retrieval system for electronic mailing lists and file archives. ListProc is intended to be easy to maintain, support and use. It can also be integrated with other data retrieval and presentation tools such as gopher, anonymous ftp, http, and WAIS.

2.1. Mailing Lists

Email discussion lists are an effective way of allowing groups of people to communicate with one another. The most elementary form of an automated mailing list is an MTA reflector. This simply takes all messages coming in to a certain address and bounces them to each member of an alias list. This can work well for small distribution lists in which mailing errors are relatively unlikely and there is no need for more advanced features.

Unfortunately, this simple scheme does not work well for larger and more complicated distribution lists. As lists grow in size, the time required to add and remove subscribers and to deal with mail problems can be tremendous. Additionally, it is often desirable to control who can and cannot post to the list, as well as to monitor postings before they go to the list.

2.2. Why ListProc?

The basic list management tasks can roughly be broken down into three main categories:

approval of list subscriptions
moderation of list postings
error mail

ListProc solves the problems of list management by separating exploiting this logical separation. For each list, ListProc allows you to define which of these tasks should be handled automatically by ListProc, and which should be passed on to an actual human for processing. ListProc offers a number of options that control how the automatic processing should proceed, so in most cases, this is sufficient. In the case that human intervention IS desirable, ListProc allows you to break up the management tasks, by giving you the flexibility to assign each of these list management tasks to a different person.

2.3. ListProc Features

As outlined above, ListProc provides a powerful and flexible mechanism for addressing a wide variety of email distribution needs. ListProc's main features are highlighted below.

Multiple Levels of Managerial and User Access

One of ListProc's key strengths is that it allows the definition of a number of different types of users, each with different capabilities. Users may add themselves to lists, send and receive messages, retrieve files from archives and set a number of parameters to define their interaction with the mailing list. List owners among other things edit or moderate lists, control subscribers or hide lists from public scrutiny.

ListProc 8.0 defines six categories of users:

Regular Users
: subscribe to lists, query the server for available lists and archives, get files and post to lists.

Subscription Managers
: are responsible for subscribing and unsubscribing users from the list. A subscription manager does not have to be subscribed to the list itself or have access to the list's archives.

Errors To Recipients
: get RFC-822 error messages and can remove subscribers from the list. Errors To Recipients do not have to be subscribed to the list or have access to the lists postings or archives.

Moderators
: approve and/or edit postings going to a moderated mailing list.

List Owners
: oversee the functions of particular lists, configure list parameters, deal with bounced mail, moderate or otherwise manage the list.

ListProc System Manager
: is responsible for overall server configuration and operation, placement of archives, monitoring logs and getting server error messages.

Not all categories need be defined at any particular ListProc site or for any particular list. The Subscription Manager, Errors-To recipient and Moderator are set to the List Owner by default. These additional options allow List Owners to distribute the responsibilities of list management.

Access Controls

ListProc allows you to flexibly define who has access to which management and subscription features. Individual lists can be set to either manager controlled or owner controlled.

Owner controlled
: gives the list owner the maximum amount of control over their list. They may add and remove subscribers, set list passwords, control list and archive traffic, and add and remove other list owners, subscription managers, and message moderators.

Manager controlled
: prevents list owners from adding or removing other owners or changing the list or archive passwords. These duties must instead be performed by the site manager.

ListProc also includes a number of settings that allow you to control whether various list commands should be available to anyone, list subscribers, or only list managers.

Email List Distribution

ListProc's most definitive feature is its ability to redistribute email sent to one address to all of the users.

USENET News - Mailing List Gateway

In addition to standard email list handling features, ListProc includes the ability to act as a gateway between a USENET news group and a mailing list. This allows users to access the information on the newsgroup or mailing list in whichever way is most convenient for them.

File and List Mail Archives

ListProc has the ability to automatically archive mail sent to a list. ListProc's sophisticated archive search and retrieval functions allow users to find the information they need from the list archives. Additionally, ListProc has the capability to create non-list archives, to allow users access to other important information. Interested users can even subscribe to particular archive files, so they can be notified whenever the file is updated, or just receive the updated file.

Email Loop and Spam Detection

ListProc automatically checks incoming messages to make sure they have not been sent to the list before. Additionally, ListProc stops email loops by making sure that incoming messages did not originate with ListProc.

Automatic Subscriber Deletion

Enabling ListProc's AUTO-DELETE-SUBSCRIBERS feature for a list causes ListProc to automatically remove subscribers whose addresses are causing bounced mail.

Configurable Messages

Many of ListProc's information and error messages are stored as simple shell scripts. This allows you to tailor ListProc's look and feel to suit the specific needs of your site.

Subscription Options

Subscribers can choose whether or not they wish to receive a confirmation that their message was sent to the list. If the list owner has enabled it, they also have the option of receiving list mail in multiple message digests, rather than as individual postings.

3. Planning your ListProc site

Before installing ListProc, it is essential have a rough idea of how it operates and how to set up your system to ensure that it can do so successfully.

3.1. ListProc Directory Structure

To simplify management, ListProc keeps all of its data files in one directory tree. The main ListProc directory (which I shall call "**LPDIR**" for the sake of convenience) contains all of the ListProc executables, as well as the server-wide configuration and data files. The **LPDIR/archives** directory and its subdirectories contain the indexes to ListProc's archives. Archived files are also kept here by default, although this can be changed for individual archives. The **LPDIR/lists** directory contains subdirectories for each of the lists in your system, which in turn contain the data files for that particular list. The names of the list directories are created simply by capitalizing the names of the corresponding lists. For example, the data and configuration files for the list "foolist" would be kept in the **LPDIR/lists/FOOLIST** directory. ListProc creates both archive directories and list directories automatically, so there is no need to add these directly.

3.2. Processing Overview

ListProc's handling of incoming messages occurs in two phases. Each ListProc list has its own incoming message folder, **LPDIR/lists/LIST-NAME/mail**. Additionally, server requests are stored in the file **LPDIR/requests**. When a message first arrives, it must be appended to the appropriate incoming message folder. This is accomplished when the local Mail Transfer Agent (MTA) invokes **catmail** through the **aliases** file (usually **/etc/aliases**). For example, incoming server mail should be sent to the email address "listproc@your.host.name." In order for this to reach ListProc, your MTA **aliases** file should contain the following line:

```
listproc: "|LPDIR/catmail -r -f"
```

(where "**LPDIR**" is of course replaced by the actual path to your ListProc directory) The arguments to **catmail** specified in the MTA **aliases** file determine the message's ultimate destination, in this case, the **LPDIR/requests** file. (See section 0 on page 84 for more information on **catmail**).

The next step is to actually process the message. ListProc's message processing is controlled by the **serverd** program. **Serverd** runs constantly, and checks for incoming server and list mail. When new mail is found, **serverd** spawns either the **listproc** (for server mail) or **list** (for list mail) programs to process it. Depending on the operation required by the incoming message, the **list** and **listproc** programs may spawn additional processes to deal with part of the processing load. Detailed descriptions of each of the ListProc executables are available in section 13, ListProc Commands and Utility

Programs (page 83). For a more thorough description of ListProc's flow of control, see section 13.7, Informational Utilities

semset

The semsetListProc Flow of Control (page 90).

3.4. Site Requirements & General Needs

ListProc runs on most UNIX based platforms. (In order for some of its more advanced internal features to work properly, it is necessary to run an operating system that includes modern file locking and memory mapping capabilities.) For some of its features, ListProc relies on the existence of a number of standard UNIX utility programs. In addition, you will need to have a working Mail Transfer Agent (MTA) that allows the execution of commands from the aliases file.

Required UNIX Utilities

ListProc makes use of the following standard UNIX utilities:

```
echo          rm            nslookup
cut           grep -i       ps
paste         uuencode      chmod
wc            compress      chown
cat           zcat          sort
tr            uncompress    uniq
md5 or sum    tar           inews
awk           egrep         UCB mail (eg mailx)
mv            tail
cp            uptime
```

These utilities should be available in the path of the server account. Furthermore, to make sure they work as expected, they should not be aliased to anything else. (For example, you should not have "**alias ls ls -F**" in your **.cshrc** file.)

MTA

In order for ListProc to receive mail properly, your machine must have a working Mail Transfer Agent (MTA) that allows the execution of commands from its aliases file.

Root Access

In order to add the appropriate mail aliases, you will need to have access to the root account on your ListProc server machine. Additionally, if you want to allow Interactive ListProc connections to the default port (372), you will need to specify the root password when using the **setup** script to install ListProc.

Disk Space

Simply installing the ListProc utilities and documentation requires less than 10 megabytes of disk space. The total amount of space you will need to reserve for ListProc depends on the amount of traffic you anticipate on your site. You will also need additional space if you plan to maintain archives of sent messages.

Memory Requirements

The amount of memory ListProc uses depends on a number of factors. Large lists and large numbers of lists will require more memory for data storage. You can control ListProc's memory usage by specifying the number of threads ListProc can use for processing mail. The more threads you allow, the greater the potential memory usage becomes.

4. Installation

ListProc is designed to be easy to install and maintain. Whenever possible, the installation and configuration steps have been automated, both to ensure that things are set up properly, and to save you time. The installation procedure is slightly different depending on whether or not you are installing ListProc for the first time, or if you are upgrading an existing list handling system. Before beginning any installation, you should read the appropriate section below.

4.1. Installing ListProc for the First Time

1) Create the "server" account

The first step is to create an account on your system for the ListProc server. By convention, this account should be called "**server**". This account will contain the ListProc binaries and data files. Most importantly, a separate server account allows effective access control for ListProc's sensitive data files. The exact procedure involved in creating the server account will depend on your system. Consult your system documentation for information on how to create user accounts.

2) Unpack the distribution files

The ListProc distribution files come in a compressed tar archive. Copy the distribution file to the server's home directory. Once there, you can uncompress and untar it as follows:

```
uncompress listproc8.0*.Z
tar -xvf listproc8.0*.tar
```

You should now have three files. **`README.INSTALL`** contains brief installation instructions. **`src.tar`** contains the actual ListProc distribution files. Finally, **`install.listproc`** is designed to unpack the files in **`src.tar`** safely in case you are reinstalling over a previous version.

Run **`install.listproc`** from the command prompt to finish unpacking the distribution files. Once this is done, you may remove **`install.listproc`** and **`src.tar`** from your system, as they are no longer needed.

3) Run "system-test"

During normal operation and configuration, ListProc relies on the existence and proper operation of the following standard UNIX utilities:

```
echo          rm            nslookup
cut           grep -i       ps
paste         uuencode      chmod
wc            compress      chown
cat           zcat          sort
```

```
tr              uncompress      uniq
md5 or sum      tar             inews
awk             egrep           UCB mail (eg mailx)
mv              tail
cp              uptime
```

The **system-test** utility checks your system to make sure all of these utilities are available and warns you if they are not. You should make sure that shell aliases are not used for any of these commands, as ListProc expects their default behaviors.

In the unlikely event that any of the required commands are not found, you should make sure that the server's PATH environment variable points to all of the standard binary directories. (Consult your system documentation or system administrator for an appropriate PATH statement for your system.) If this does not solve the problem, you will need to install the missing utilities.

4) Run the "setup" script.

The **setup** script does a number of things. First, if the LPDIR environment variable is set, it copies all of the distribution files to the directory specified. Second, it ensures that the file permissions are properly set for each of the system binaries. In particular, it will set the "setuid" bit for both **catmail** and **serverd**. Setup also gives you the opportunity to have **serverd** owned by root, if you want to allow ILP connections to a privileged port. Finally, **setup** will rewrite some of ListProc's data files to reflect the path name to the server account on your system. This is extremely important, as it ensures that ListProc will be able to find its help, archive, and news files.

5) Setup the system mail alias

In order for your ListProc server to receive and process mail, you must set up a system-wide alias for it. The aliases file is usually **`/etc/aliases`** or **`/etc/mail/aliases`**, although it can reside in other places as well. (Note: root access will be needed to modify the aliases file.)

The server alias is a pipe to a process called **catmail**. (A detailed description of **catmail** is provided in section 0 on page 84.) The server alias should be as follows:

```
ListProc: "|/LPDIR/catmail -r -f"
```

Note that "LPDIR" should be replaced with the full path name to your ListProc installation directory. For example, if ListProc is installed in the **`/home/server`** directory, your aliases file should contain the following line:

```
ListProc: "|/home/server/catmail -r -f"
```

Do not forget to run the your system's **newaliases** command to rebuild the alias database after you edit the aliases file.

Note: Your mailer must allow you to launch programs.
As a security precaution, some versions of **sendmail** do not allow programs to be executed by the mailer. They implement this via some form of restricted shell. It is important that you or your sysadmin familiarize yourself with your particular implementation. A detailed explanation of mailers is beyond the scope of this manual.

For **sendmail** documentation, see **sendmail** by Bryan Costales with Eric Allman and Neil Rickert.

6) Set the system environment variables

ListProc also looks at several environment variables in order to determine aspects of its behavior.

LPDIR
This variable defines the default directory for ListProc. All of the main server programs use this environment variable to determine where to find ListProc's data files and auxiliary commands. Hence, it is essential that this environment variable be set correctly if the server is to run correctly.

MD5
If this variable is defined, ListProc will use the **MD5** algorithm for checksums instead of the system call **sum**. MD5 provides greater width in calculating the checksums used in some of ListProc's loop detection procedures. MD5 is available via anonymous ftp server at info.cren.net in /pub/software/MD5. Please note that MD5 must be in the server's search path for ListProc to be able to find and use it.

ULISTPROC_UMASK
This defines the file permissions ListProc uses when creating and modifying files. **It is critical for file security that this be set to 077!**

ULISTPROC_ARCHIVES_UMASK
This variable establishes separate file permissions for the ListProc archives directories and files. This mask affects only the archives. This needs to be other than 700 if you plan to allow anonymous ftp/gopher or other server access to the archives. For example, a mask of 022 creates archives with permissions 644. This will allow anonymous gopher, WAIS, and ftp users to access the archives.

ULISTPROC_ACCOUNTING_PROGRAM
This specifies the path to an accounting program to be used each time messages are sent out from ListProc. The arguments passed to the accounting program are as follows:

the list alias
the value of LPDIR
the total number of regular mail recipients (including peers)
the total number of digest recipients
the total number of newsgroup postings
the filename that contains the headers of the message distributed
the filename that contains the body of the message distributed

To ensure that all of the appropriate environment variables are set properly each time you start the system, you should set them in your shell's initialization file. This commands in this file will be executed each time you log in to the server account. If you use **su** to change to the server account, make sure you use the "**-**" argument as below:

```
unix% su - server
```

This will ensure that the shell is a login shell, and hence that the shell initialization files will be read.

The name of the shell initialization file and the specific commands you should use depend on the type of shell you use for the server account. Examples for the two most common shell families are listed below:

The C-shell
The shell initialization file used by the C-shell and its variants (**csh**, and **tcsh**) is **.cshrc**. You should add the following lines to the **.cshrc** file in the server's home directory. (If there is no **.cshrc**, you should create one.)

```
setenv LPDIR path-to-server-directory
setenv ULISTPROC_UMASK 077
setenv ULISTPROC_ARCHIVE_UMASK 022
# Uncomment the following if you want ListProc to use MD5
#setenv MD5 1

unalias ls rm ....
```

The Bourne Shell
If you are using a variant of the Bourne shell (**sh**, **bash**, **ksh**, **zsh**, **ash**, etc.), you should add the following lines to the **.profile** file in the server's home directory:

```
LPDIR=path-to-server-directory;  export LPDIR
ULISTPROC_UMASK=077;  export ULISTPROC_UMASK
ULISTPROC_ARCHIVE_UMASK=022;  export
ULISTPROC_ARCHIVE_UMASK
# Uncomment the following if you want ListProc to use MD5
#MD5=1;  export MD5
```

7) Customize the ListProc configuration files to suit your needs.

At this point, the basic installation is complete. Before you can start the system, you will need to modify the **LPDIR/config** file. For more information about modifying the **config** file, see section 5 on page 19.

4.2. Upgrading from ListProc 7.x

The procedure for upgrading from a previous version of ListProc is nearly identical to that used in a fresh installation. Although steps 1, 3, 5, and 6 should have been done when you installed version 7, you may wish to check again to make sure that everything is set up properly. A few additional steps must be performed when upgrading. These are listed below. The numbers indicate where these steps should be inserted into the above sequence.

0) Stop incoming mail to your server machine.

While you are upgrading ListProc, there will be a short period of time during which the file permissions on **catmail** may not be set correctly. This would cause incoming list and server mail will bounce with an "unknown mailer" or similar error. To prevent this, it is

a good idea to set up your system to return a more friendly error, that will cause the sender to simply re-queue the mail for later delivery.

The best way to solve this problem is to simply stop your SMTP server (i.e. **sendmail**). This will disrupt all mail to and from your site. No incoming mail will be lost, however, since sites trying to send you mail should simply queue it for later delivery. Consult your system documentation for information on how to stop your SMTP server.

If it is not feasible to stop your ListProc machine's SMTP server,

1a) Stop any existing server processes.

Before upgrading ListProc, you should stop any existing ListProc processes with the "**stop**" command. While logged in to the server account, issue the following command:

```
unix% cd $LPDIR
unix% ./stop
```

(Note: it is important to specify the path to the **stop** command as "stop" is a built-in key word for some UNIX shells.) Answer "Y" to each prompt to kill the existing server processes.

1b) Backup your existing server tree.

Although the upgrade procedure is designed to be non-destructive, it is always a good idea to make a backup of your current system before proceeding.

2a) Decide whether to use new or existing data files.

When you run the **install.listproc** command in an existing ListProc directory, your original configuration, archives, and lists files will be renamed to **`Ofilename`** to make sure they are not overwritten when the files are extracted from **`src.tar`**. When all of the files have been extracted, the new versions are renamed to **`Nfilename`**, and the **`Ofilename`** files are restored to their original names. Before continuing to step 3 of the installation, you should decide whether you wish to use the old or the new versions of these. The files and directories that are protected in this manner are as follows:

```
help                    .ignored
archives                .aliases
config                  welcome.live
owners                  welcome.global
unwanted.hosts          signoff.global
```

For **`archive`** and **`lists`** directories, you will almost certainly want to use the original versions, as these still contain all of the information from your previous installation. You should also definitely keep the **`config`**, **`owners`**, **`.ignored`** and **`.aliases`** files, as these contain important information about the setup of your site.

The decision whether or not to use the new or the old versions of the other files is entirely up to you. Often times (in particular with the system **`config`** file) it is useful to keep the new versions around for reference purposes. Unless you have customized the help, welcome, or signoff files, we recommend that you replace these with the newer versions included in the distribution.

8) Force ListProc to send any partial digests.

The format of list digests has changed slightly between ListProc versions 7 and 8. To make sure the first digests aren't saved with incorrect formats, you should force ListProc to send out all existing partial digests *before* starting ListProc 8.0. To accomplish this, ListProc 8.0 includes the **force_digests** command. After completing step 7 in the previous section, run **force_digests** from the server directory. It is important to make sure that ListProc has been completely shut down before you do this. (It should have already been shut down in step 1a.)

4.3. Upgrading from ListProc 6.0

In order to upgrade successfully, you should first follow the steps in the previous section to install over an existing ListProc directory. The formats of many of the data files have changed between 6.0 and 8.0. CREN provides two utilities that help to update these.

7a)

After the setup script finishes, you must now do some general housekeeping. We have provided two scripts to help you do this. These scripts actually make changes in each list's subdirectory $LPDIR/lists/LIST-ALIAS/Ö and generate a new config file for you.

tconfig
: This is a bourne shell script which must be run as the ListProc's uid. This version of ListProc requires a config file in each lists' directory. This file contains list-specific configuration information that is under the control of the list owner. This script simply "touches" this file in each list sub directory so that the server can run the first time through.

convert.pl
: This is a perl script. It takes your owners file and your original config file and generates two files: **listname-init** and **listname.alias**. The first file is a config command (see Config below) which is used to configure the list to the old parameters. It is ready to send to ListProc in this form. The second file is the system aliases for the mailing list which must replace the old aliases. (This part requires root access). Convert will also append a correct *list* command to the end of the new config file (if you use convert more than once on the same list, you must edit the config file and remove the duplicate list statement). The convert script has many variables which must be customized to your site. The comments at the beginning of the script list the variables that you must change.

After you remove all the old aliases from the system alias file and add the new ones, you must run the newaliases command. (This is a system command, not a ListProc command and it requires root access).

Start ListProc. ListProc must be running when you send it all the listname.init files that you created during the upgrade process. These files must be sent, for this initial step, from the ListProc's manager's account. You can do this via e-mail or using the interactive client (see ILP). If you use e-mail, you must restart the system mailer (root access) before this step, otherwise you can restart the system mailer after this step.

Notes:
ListProc 6 allowed certain characters in the list password field which are not allowed under 7.2 and above. Try to avoid using the quotation mark, apostrophe, and the comma (" ' ,).

Unless your site is running the script **re_from**, ListProc 6.0 and below use the envelope address (the "From " header) instead of the "**From:**" header. In order to conform to the relevant RFCs, ListProc 7.2 and above take the user's address from the "From:" header. Unfortunately, this means that some users from previous versions of ListProc may not be recognized after the upgrade.

4.4. Upgrading from versions prior to 6.0

To convert from ListProc's before 6.0, you must first upgrade to ListProc 6.0c. You can obtain a copy of ListProc 6.0x via anonymous ftp from **`cs-ftp.bu.edu`**. In the transition from 5.5 to 6, the structure of the subscribers file underwent fundamental changes which first must be addressed. Contact info@cren.net for further information.

4.5. Converting from VM Listserv

If you are converting from VM Listserv, two conversion tools are available in the support directory of the ListProc distribution. These tools are:

lsv_conv_list.pl
This utility converts a VM Listserv subscribers file into ListProc format. It preserves most of the user options although some user options have no equivalent.

lsv_conv_logs.pl
This utility converts VM Listserv logs into a format suitable for ListProc archives. It changes the logs to UNIX mbox format and places them, using **farch**, in the archive directory.

Both tools are perl scripts and require that you first ftp (or otherwise transfer) the VM Listserv list header and subscribers file to the host on which ListProc resides.

4.6. Installing the Man Pages

The **`doc`** subdirectory of the ListProc distribution includes a number of manual pages intended to provide quick reference to ListProc's commands. Each man page comes in two forms. The files ending with ".nr" can be viewed directly with "more". The files ending with ".1" should be viewed with "man". To install the ".1" files, simply copy them to one of your system's manual page directories. For example, to install them under /usr/local/man, you should do the following:

```
unix% su root
root's Password: (type the root password here)
```

```
cp /home/server/doc/*.1 /usr/local/man/man1
```

Once this is done, you need only type “man command-name” to access the manual pages.

5. Basic Configuration - The Site Config File

At this point the basic system is installed and we are ready to set system configuration parameters. In all likelihood, the config file will only need to be manually edited during initial installation and whenever there are changes to related components of your system. (For example, if you upgrade your operating system or your mailer, you will probably have to revisit the config file.) The system comes with a default config file that you can customize to meet the particular needs of your site. (See Appendix D for an example.)

5.1. Config File Directives

The config file contains a number of options and setting that define the operation of your site. These are described below. Text that should be copied exactly as it appears is written in **`bold typewriter font`**, while variable arguments are shown in ***`bold italicized typewriter font`***. Optional arguments are enclosed in square brackets ([ ]). Valid config file options are as follows:

`organization` ***`name-of-your-organization`***

This line defines the Organization that users will see on interactive login to the server. It is a free text string and does not need to be quoted.

`multiple_recipients` ***`number-of-recipients`***

This defines how many recipients ListProc sends to your mail transfer agent (MTA) per connection. In general, the higher this number is, the more efficient your mail transfers will be. The most appropriate value depend on your specific system and mailer, however. We recommend starting with a value of 25, which works well with sendmail. With **zmailer**, a value of 100 is acceptable. If no **multiple_recipients** line appears in the config file (or if it has been commented out), ListProc sends only one message per MTA connection.

`list` ***`list-name list-address`*** **`[`*** ***`list-owner`*** **`] [`*** ***`list-password`*** **`] [-D] [-e]`**

This directive defines a particular mailing list. Starting with ListProc version 7.2, there is no need to edit this field, as it is added automatically by the **initialize** command. The list directive is explained here to make the site manager aware of its syntax and debug any problems that might arise. This line should only be directly manipulated when you are removing a list from your site, in which case the entire line should be deleted. (See section 7.3, “Removing a List,” on page 57.)

list-name
the name of the mailing list

list-address
the fully qualified e-mail address of the owner

list-owner*
the fully qualified address of one of the list owners or an alias which points to the list owners.

list-password*
The initial password for the list owners to use to configure and access the list

***** With ListProc 7.X and beyond, the *list-owner* and *list-password* fields are no longer used with the **list** config file directive, as List owner information is stored in the **`LPDIR/owners`** file, and the list password is stored in **`LPDIR/lists/LISTNAME/config`**. These fields are maintained only for backward compatibility. In both cases, these values should be set with the INITIALIZE and CONFIGURATION commands (explained below), rather than through direct editing of the data files.

With the **initialize** command you no longer need to specify the list directive in the config file. We explain it here to help site managers understand the config file and backwards compatibility.

-e
Echo Reports to the screen. If you have a modern syslog facility, this flag should not be used. If syslog is enabled, the flag is ignored, and no echoing to the screen will occur.

-D
This will tells the "list" process for that list to record all MTA transactions in the files **`LPDIR/lists/LISTNAME/sent`** and **`LPDIR/lists/LISTNAME/received`**.

**`global-update-server *address time***`
This option defines the server to which to send daily updates of the published lists on your site. These lists will be available

address
refers to the address of the server

time
The time at which to compile and send a list of published lists to the global server.

Example:
global-update-server global-update-server@listproc.listproc.net 9:00

`global-query-server hostname port`
If there is no local remote list file, ListProc sends all queries which it cannot resolve locally to a global query server.

Example
To set the global query server to listproc.listproc.net, with the default connection port, you should use the following line:

```
global-query-server listproc.listproc.net 372
```

`server *email domain* [-b *request*] [-r *command*] [-d *command*]`
This command defines the e-mail address of the server.

email
The e-mail address of the server. For example, the email address for CREN's ListProc server is "listproc@list.cren.net".

domain
The full Internet hostname of the machine running ListProc.

-b *request*
This flag instructs the server to batch all requests to off peak hours. For example, if getting statistics puts a large load on your system, you can postpone all statistics requests to non-prime hours with the following:

```
server ListProc@host.domain host.domain -b statistics
```

Defining peak/off peak hours is discussed later in this section.

-r *command*
This flag allows the site administrator to disable a particular command while the system load is above a configured number. For example:

```
server ListProc@host.domain host.domain -r statistics
```

This disables the *statistics* command for the entire site if the number of users logged in exceeds the number specified in the restriction parameter (defined below).

-d *command*
This flag removes a particular command from the available set for use. It effectively disables the command on this server. For example, to disable the *statistics* command, use following parameter:

```
server ListProc@host.domain host.domain -d statistics
```

`threads system_wide` ***`number`***

The **threads** directive controls the total number of **list**, **listproc**, and **pqueue** processes that ListProc can spawn at any given time. The ***number*** parameter can be between 1 and 255, inclusive. If only one thread is used, ListProc's processing sequence will be identical to that used in version 7.x. Specifying more than one thread allows ListProc to process error mail, server requests, and mail for multiple lists concurrently. Individual lists may be configured to use multiple threads for sending outgoing mail. (For more information on the setting the number of threads for a given list, see page 54.)

`unix_cmd` ***`list-name password alias 'unix-command [args]' #comment`***

This request may be used to allow a particular list's subscribers to execute the above defined UNIX command. Since no security checks are made except to ensure that the user arguments do not contain: ‘, |, :, < or >, we encourage you to review carefully the functioning of this argument. It is included for backwards compatibility with ListProc 6.0, but we discourage using it.

list-name
The name of the list for which the command is to be defined.

password
The password required for users to access this particular command. List users who should have access to this command must be given this password. List owners and site managers may use the list and site passwords, respectively.

alias
The command name users will use to execute the command.

***unix-command* [*args*]**
The actual UNIX command to be run by ListProc. The optional arguments, may be literal arguments to the command as well as the special sequences $*n*, where *n* is a digit from 1 - 9. Their meaning is that of the bourne shell and they will be substituted by the user's actual arguments.

Example
To swap the first and second user arguments, you would use the following line:

```
unix_cmd foo-list foo-pw swap '/bin/echo $2 $1' #Syntax
swap a1 a2
```

To access this command, a user would submit a request such as this:

```
run foo-lost foo-pw swap arg1 arg2.
```

The comment field is used in response to the user command

```
run listname
```

which queries for run commands available.

Again, since **NO SECURITY CHECKS ARE MADE FOR THE RUN COMMAND**, use it with extreme caution. You are allowing a remote user to spawn a process on your system via email with this command. CREN does not recommend its use. The run command can be disabled via the -d flag in the server directive (see above) with a config line such as:

```
server listproc@host.domain -d run
```

batch *starthour endhour*
This command defines the hours (inclusively) between which requests will be batched. This allows you to limit the amount of time ListProc spends on certain user requests during your system's hours of peak activity.

starthour
The beginning of the peak period.

endhour
End of the peak period.

Example
To batch requests between 8 a.m. and 8 p.m., you would use the following line:

```
batch 8 20
```

The commands that should be batched are defined in the server line of the config file. (See the section on the server directive,

```
server listproc@host.domain host.domain -b statistics
```

Any user who sends a statistics request to ListProc during the pre-defined prime hours, will receive a message telling them that the request will be batched for processing at a later time.

serverd [-l *load*] [-i *duration*] [-p *port*] [-c *max-cons*] [-e]

This command defines what flags are passed to the server deamon, the principle ListProc deamon. The available flags are:

-l *load*
Postpone filling requests if the system load goes above that specified on the command line.

-i *duration*
Go interactive and listen for incoming connections. If you want ListProc to listen on a priviliged port (such as the default 372) **serverd** needs to be owned by root and setuid. The duration field specifies the maximum number of seconds an interactive session will last. In order to go interactive, in addition to the -i flag, you need the following entry in your /etc/services file:

```
ulistproc   372/tcp
```

Note: AIX users should use *smit* to modify system files.

-p *port*
Specify the listening port. The default is 372.

-c *max-cons*
Specify the maximum number of interactive connections, the default is 5.

-e
Echo messages to the screen. Not valid if syslog is defined.

Example
A typical serverd line might look like this:

```
serverd -l 5 -i 1800 -c 10
```

This will postpone filling requests if the system load rises above 5, go interactive and allow interactive users a maximum of 1800 seconds (20 minutes) per session, and allow a maximum of 10 interactive users.

restriction *max*

This command specifies that certain requests will not be satisfied when the number of users logged into the system rises above ***max***. This number refers to all users, not just interactive ListProc sessions. You specify the commands that are disabled using the -r flag with the **server** directive (see page 20).

Example
If you want to restrict statistics requests when more than 50 users are logged onto your system, you could use the following server and restriction lines in your config file:

```
restrictions 50
server listproc@host.domain host.domain -r statistics
```

These two lines can appear in either order.

manager *manager-email-address*

This directive defines the e-mail address of the ListProc site manager. If you require multiple managers, you can set ***manager-email-address*** to an alias, rather than a specific email address.

password *manager-password*

The manager password is defined here. In the interest of security, this should not be the same as any user password.

comment server Site Comment

This defines a comment field for all outgoing postings and messages. This field will appear in the **From:** line all email the server sends out in response to user requests. For example, to set the site comment to "CREN ListProc Version 8.0," you would use the following line:

```
comment server CREN ListProc Version 8.0
```

frequency *list-delay* [*loop-delay*]

This directive controls how often ListProc checks incoming list and server email. ListProc handles incoming mail as follows:

```
process mail for list 1
process server mail
wait list-delay seconds
process mail for list 2
process server mail
wait list-delay seconds
....
process mail for final list
process server mail
wait list-delay seconds
wait loop-delay seconds
repeat
```

Notice that the last step is to wait for both ***list-delay*** and ***loop-delay*** seconds. **For this reason, *loop-delay* is considered to be zero unless *list-delay* is zero.**

Example
To set ListProc to pause for 10 seconds after each time it completes one of the list-server mail checking pairs, use the following line in your config file:

```
frequency 10
```

Example
To have ListProc pause for one minute each time it cycles through all of the lists, use the following line:

```
frequency 0 60
```

These two mechanism allow you to tailor ListProc's processing, depending on your the load on your server. If have only a few lists, using ***list-delay*** may be a good way to keep ListProc from unnecessarily cycling through the lists, and checking for new mail. As the number of lists on your server increases, however, using ***loop-delay*** becomes more desirable, as the mail for each list is processed more frequently.

limit message ***max-bytes***
limit files ***max-bytes***

The **limit** directive allows the site manager to control the maximum size of messages sent to ListProc, and to limit the size of archives ListProc sends out to users.

message
When the **message** option is used, messages longer than ***max-bytes*** bytes will be rejected. A notification is sent to the sender along with the first few lines of his/her original message for reference. With the advent of MIME, messages of 1+ Megabytes are not uncommon. If your users want to send and receive multimedia documents on a mailing list, you must set this to a higher number. (One to two megabytes might be more appropriate.) If no "limit message" line appears in the config file, ListProc will accept incoming messages of any size.

files
The **files** option is used to limit the size of the archive files that ListProc will mail out to users. All files greater than ***max-bytes*** will be automatically split up into subparts prior to mailing. When a user requests a global list of lists it too will be split up if it exceeds this limit. By default, there is no limit on the size of archive files.

precedence ***mail-precedence***

This specifies the string to be used in the **Precedence:** field on outgoing list mail. It is usually set to **bulk**, **junk**, **first-class** or **none**. Vacation programs are *supposed* to not respond to bulk mail if they receive it. It is hoped that most vacation programs are well behaved in this regard. (It has also been reported that certain DECNET gateways will not forward bulk mail.) We recommend setting this to **bulk**. If you comment out this option, ListProc will not use a **Precedence:** field. This command is correctly used as follows:

```
precedence bulk
```

ucb_mail ***path-to-mailer***

This defines a path to your BSD mailer if it is available on your system. It is usually /bin/mail, /bin/mailx, /usr/ucb/mail or /usr/ucb/mailx.

fax ***fax_program options***

This defines the fax program to use for fax request (if any). The fax program must be able to accept the file to fax from its standard input; otherwise you will need a script to use the fax program appropriately. If this directive is left off or commented out, fax requests are turned off.

To use this feature, you must have a second party fax driver that can receive input from ListProc and a modem that can send faxes.

```
header list-name|* {
      [!]header-line1
      [[!]header-line2]
      ....
}
```

The **header** was originally used to specify which headers should be preserved for all incoming mail messages. ListProc versions 7.2 and above automatically retain all headers, so this directive is no longer needed for this purpose. Starting with version 8.0, the capabilities of the **header** directive have been expanded to allow sites to explicate specify header lines that should *not* be saved.

Although ListProc attempts not to alter the headers on incoming messages, there are a few occasions in which this is necessary:

1) If you have your list set to always set reply to the list, then any user supplied **Reply-To:** header will of course be lost.

2) The original **To:** and **CC:** headers are converted into **X-To:** and **X-cc:** to let the list know if there were originally other recipients of this message as well.

***list-name* | ***
This specifies the name of the list whose header lines should be checked. An asterisk in place of the list name means that all lists should be checked. If multiple **header** lines exist for the same list, the functionality is the same as if all of the header lines had been defined in the same **header** line in the config file.

[!]*header-line*
Headers are specified within braces ({ }), and separated by white space (tabs, spaces, and new lines). Individual header are described by regular expressions, so you have a good deal of flexibility in defining which headers should be preserved or discarded. Note that these regular expressions are compared only to the header itself (the part before and including the colon), *not* to the content portion of the header line. Also, since the regular expressions are delimited by white space, you cannot use white space characters when describing a header. This should not cause any difficulties, as the headers themselves should not include white space characters.

Example:
The following prevents the propagation of headers that cause loops and error messages with auto-responders:

```
header * {
      !Registered-Mail-Reply-Requested-By:
      !X-Confirm-Reading-To:
}
```

Example:
To preserve all **Errors-To** and **X-** headers on the list “foolist”, you could use the following:

```
header foolist {
      Errors-To:
      X-.*
}
```

Note:
There are several types of headers that the **header** config file directive will not effect. First, the **header** directive cannot be used to override ListProc's re-writing of the **To** and **CC** headers. If you want these headers to be copied directly, you must set your list to REFLECTOR. (See page 54 for a description of the REFLECTOR setting.)

The following header lines will be preserved regardless of the **header** directive:

all MIME headers	Message-Id:
Approved:	Reply-To:
Archive-Name:	Resent-From:
Control:	Resent-Message-Id:
Date:	Resent-Sender:
From:	Sender:

Caution:
When used to control how ListProc handles the headers on a specific list, the **header** directive *must* appear after the list has been defined with the **list** directive.

```
option gate_mail|post_mail
option ignore_invalid_requests
option relaxed_syntax
option multiple_listprocs
option readonly_subscribers_files
```

The **option** directive allows the site manager to set some of ListProc's general options. Multiple **option** lines are permissible in the config file.

gate_mail
This option forces the system to gate messages to news using the gateways e-mail address. Some popular packages are **mail2news** and **gatemail**, both widely available on the Internet. ListProc does not directly use these but simply sends mail to the specified address for further action. (See the subsection on News-List gateways on page 81.)

post_mail
This option forces the system to post messages to news groups (via inews) using the groups' names (e.g. misc.test). To use this, make sure that inews resides in /usr/lib/news, or that the config directive **inews** has been defined.

ignore_invalid_requests
This option instructs the server to ignore invalid commands in incoming email. When this option is turned off (the default) the system stops processing requests when it encounter the first invalid one, at which point the users gets an error message and all subsequent requests are flushed. With this option enabled, all valid lines requests are processed, and no error message is generated.

relaxed_syntax
This option instructs the server to ignore all non essential parameters to a valid listproc command. It will execute the command and not give an error. For example, consider the following user command:

```
unsubscribe java-update Joe Java
```

Normally, this will generate an error message, since the **unsubscribe** command requires only the list name as a parameter. If the **relaxed_syntax** option is specified, however, ListProc will ignore the extra arguments and unsubscribe the user without generating an error message.

multiple_listprocs
This option is for sites running two or more ListProc servers who want them to behave as a seamless unit. To accomplish this, the **`LPDIR/remote.lists`** file for each server should contain the lists local to each of the other servers. With this config file option, ListProc will not send the standard notification to users, " ÖThe list xxx, is not local, your request has been forwardedÖ" Thus, need never know that his/her request was being forwarded to another server. In this way, one address can be set up for all administrative requests to listproc.

readonly_subscribers_files
This option disables the PUTÖSUBSCRIBERS by owners. This will prevent owners from putting back a subscribers file which is not syntactically correct and causing ListProc errors. Owners will be forced to use the ADD and DELETE commands to modify the subscribers file. They can still edit the file but in read only mode.

`inews` ***`path`***
This directive tells ListProc where to find the inews program for news - mail postings. ListProc will look in /usr/lib/inews by default.

`mailmethod` ***`method-name`*** **`[`*** ***`arguments`*** **`]`**
This option allows the site manager to define the method through which mail will be delivered. The recommended method is **system**, as it provides a unified approach to sending mail and has been tested on a wide variety of systems. In addition, the full set of the SMTP protocol is implemented, it allows list-owners and the manager to be notified of mail delivery problems. Also, with this method, mail is queued when it cannot be delivered. This method requires host TCP/IP connectivity and Internet support. An SMTP mail agent must exist on your system.

If you need to institute any of the other available mail methods, please talk to the CREN off line.

By using the **system** mail method, if the SMTP server is down or goes down while ListProc is distributing mail, the mail gets queued. The system manager will be notified and can take action. All mail is queued in **`LPDIR/mqueue`** and the queued files have numeric names, 1001, 1002. The manager should check the files before sending them out because queued mail may indicate problems other than the SMTP server going down, (i.e. malformed headers). To send the queue once the problem has been resolved, the

manager must run the **queued** included with the distribution. (For more information on **queued** see the man page, **`LPDIR/doc/queued.1`**.)

telnet
To be used when **system** is inappropriate for your system and telnet is available on your system. The behavior is similar.

env-var
Usually followed by <logname>/bin/rmail, or <logname>/usr/lib/sendmail -ha (-ha is required only to be used when the previous methods are inappropriate.)

sendmail
The **sendmail** method sends the messages directly to your MTA. Note that there are a number of limitations of this method. First, in order for ListProc to be able to set the "From" header in this case, the server account must be a trusted user of sendmail. (See the section on mailers on page 81 for more information.)

`sendmail` ***`path -flags`***
This specifies the sendmail command that will be used if **mailmethod** is set to "sendmail".

`deliver_files hourly|daily` ***`time`***
`deliver_files weekly` ***`day-of-week`***
`deliver_files monthly`
This directive controls the frequency and time that files are delivered to subscribers. This applies only if the are accessible through AFD/FUI.

Examples of correct usage are shown below. (You should use only one such line in your config file.)

```
deliver_files hourly 30
deliver_files daily 8:30
deliver_files weekly Monday
deliver_files monthly
```

`mta_host [`*`host`*`] [`*`port`*`]`
This directive defines the MTA (mail transfer agent) host and the connection port for sending outgoing mail. The default host is localhost; the default port is 25.

`helo_arg` ***`arg`***
The helo_arg directive defines the HELO argument for the SMTP transaction to the mta_host. Mail Transfer Agents such as **zmailer** complain when HELO argument does not match their view of the network. This allows a site to tailor that argument. The default value of *arg* is localhost.

`syslog` ***`facility`***
This allows a site to turn on syslog reporting. The *facility* argument can be any one of the following:

```
LOG_USER
LOG_DAEMON
LOG_MAIL
```

```
LOG_LOCAL0
LOG_LOCAL1
...
LOG_LOCAL7
```

In addition to the line in your ListProc config file, you will need to define the log facility in the **/etc/syslogd.conf** file. ListProc uses the log levels info and debug.

Turning on syslog disables reporting.

Example:
To log to a file called **/var/log/list-log** using the LOCAL2 facility, you should use following line in your config file.

```
syslog  LOG_LOCAL2
```

The appropriate syntax for your **/etc/syslogd.conf** file may vary slightly from system to system. For BSD 4.3 syslog systems, the following line is adequate:

```
local2.info;local2.debug      /var/log/list-log
```

mailer_daemon *'addr1|addr2Ö'*

ListProc defines a number of suspicious addresses that are not allowed to post to lists. The **mailer_daemon** directive allows the site administrator to specify additional blackout addresses. The argument is simply a large regular expression. (For an explanation of ListProc regular expressions, see Chapter 18.) Individual blackout addresses should be separated by "|".

When ListProc processes the config file, the **mailer_daemon** expression is concatenated to ListProc's built in set of blackout addresses. An extra "|" is added between the two regular expressions, to make sure the resulting regular expression is properly formed. For example, if the built in blackout addresses were defined by "r_default" and the user defined blackout addresses by "r_user," ListProc would check all incoming messages to see if the were sent by someone who matched the following regular expression:

```
r_default|r_user
```

ListProc's hard-coded blackout addresses are listed below. For ease of readability, they are listed as separate regular expressions, rather than as one, "|" separated regular expression:

```
<MAILER&~.+@.*MAILER.+>
SMTP@
<DA?EMON&~DEMON\.CO\.UK>
POST.*MAST
 WPUSER
\$EMD
VMMAIL
MAIL.*SYSTEM
-MAISER-
MAIL.+AGENT
TCPMAIL
BITMAIL
```

```
MAILMAN
MAIL_SYSTEM
^LISTSERV
^LISTPROC
^SERVER
^LISTMAN
^CRENLIST
```

Example
The mailer_daemon line included in ListProc's default configuration file is as follows:

Note that **root** is in this list. Some sites which use ListProc have automated servers that post mail to a mailing list as uid **root**. These sites required the flexibility of allowing the root account to post to a mailing list. If you do not want **root** to be able to post, leave this in.

Example
Since ListProc internally deals with blackout addresses as one big regular expression, you can override ListProc's built in blackout addresses through judicious use of negated sub-expressions in the **mailer_daemon** config file line. For the sake of simplicity, suppose the built in blackout addresses were defined by the single regular expression "foo|bar|bat". The following would exclude "frazzle" but allow "foo":

```
mailer_daemon 'fe|fi|fo&~foo'
```

This works because the "|" binds more tightly than "&". Hence, the resulting internal expression,

```
foo|bar|bat|fe|fi|fo&~foo
```

is treated identically to

```
<foo|bar|bat|fe|fi|fo>&~foo
```

Note that it is essential that all of the "&~expression" parts of the **mailer_daemon** expression occur at the end, to prevent fouling up anything else. In the above example, if we had instead done this

```
mailer_daemon '&~foo|fe|fi|fo'
```

The resulting expression would have been interpreted as follows:

```
<foo|bar|bat>&<~foo|fe|fi|fo>
```

This would only match expressions that contained one of "foo," "bar," and "bat," *and* either didn't include "foo" or did include one of "fe," "fi," and "fo." Hence "fe" which would have been rejected the first time will *not* be rejected because it doesn't also match "foo|bar|bat."

susp_subject *'subj1|subj2Ö'*

This option allows the site administrator to specify suspicious subjects. As with **mailer_daemon**, the regular expression specified here is concatenated with one built in to ListProc. Messages with Subject: lines that match this regular expression will be rejected, and the manager will be notified.

The components of the built in regular expression are included below. For clarity, they are listed separately, rather than as one regular expression. Also, the tab characters below have been replaced with "*TAB*":

```
DELIVERY[ TAB]+ERROR
DELIVERY[ TAB]+REPORT
DELIVERY[ TAB]+PROBLEM
WARNING[ TAB]+FROM[ TAB]+UUCP
USER[ TAB]+UNKNOWN
UNDELIVER.+[ TAB]MAIL
UNDELIVER.+[ TAB]MESSAGE
PROBLEMS.+DELIVERING.+MAIL
PROBLEMS.+DELIVERING.+MESSAGE
CAN[ TAB]*['NO]*T.+DELIVER.+MAIL
UNABLE.+DELIVER.+MAIL
UNABLE.+DELIVER.+MESSAGE
MAIL.+NOT.+DELIVERABLE
MESSAGE.+NOT.+DELIVERABLE
FAILED[ TAB]+MAIL
FAILED[ TAB]+MESSAGE
MAIL[ TAB]+FAILED
MAIL[ TAB]+RETURNED
RETURNED[ TAB]+MAIL
MAIL.*[ TAB]ERROR
MAIL[ TAB]+RECEIVED
MESSAGE[ TAB]+RECEIVED
MESSAGE[ TAB]+DELIVER
MAIL[ TAB]+DELIVER
INTERCEPTED[ TAB]+MAIL
WAITING[ TAB]+MAIL
READ[ TAB]+RECEIPT
RECEIPT[ TAB]+NOTIFICATION
STATUS.+SIGNAL[ TAB]+[0-9]+
^ERROR[ TAB]+CONDITION[ TAB]+RE:
^NO[ TAB]+REQUESTS[ TAB]+FOUND
AUTO[ TAB]+REPLY
AUTOMATIC[ TAB]+REPLY
AUTOMATICALLY[ TAB]+GENERATED
AUTOANSWERED
WAITING.+MAIL
ON[ TAB]+VACATION
VIA[ TAB]+VACATION
CONCIERGE[ TAB]+NOTICE
AWAY[ TAB]+FROM.+MAIL
CAN[ TAB]*['NO]*T.+ANSWER
CAN[ TAB]*['NO]*T.+REPLY
OUT[ TAB]+OF[ TAB]+TOWN
OUT[ TAB]+OF[ TAB]+OFFICE
```

sense_requests *'req1|req2Ö'*

This directive allows a site manager to check incoming list messages to see if they contain ListProc commands. If the first line of the message matches this regular expression, the user will get an error message stating that a request was sent to the list

and should instead be sent to the ListProc address. If any part of the regular expression is enclosed in parenthesis, the part that matched the inside of the parenthesis will be included in the error message, so the user will know which request was sensed. The default config file contains a comprehensive list of requests.

Example
The following would look for "lists" requests in the first line of incoming list messages:

```
sense_requests '^[ TAB]*(LISTS?)[ TAB]'
```

The parentheses around "LISTS?" specify that the part of the line that matches that regular expression (i.e. either "list" or "lists") should be included in the error message, to tell the user which request was sensed.

ignore_requests *'req1|req2Ö'*

This directive allows the site manager to define requests that can be safely ignored and should not generate an error message. The argument is a regular expression.

Example
The following line ignores a couple of common requests:

```
ignore_requests 'this is a test|please ignore'
```

error_analysis *level search-period*

ListProc performs error analysis on SMTP bounced mail. It attempts to determine the type of error and the originating user. ListProc's response depends on the type of error it detects. For apparently unrecoverable errors, such as bounced mail, ListProc will auto-delete the offending user. ListProc will postpone mail to users when the bounce is of an interim type such as "Disk Quota Exceeded ."

ListProc will notify the list owner of all auto-deletes and auto-postpones.

level
This can be either 1 or 9. Level 9 analysis will auto-delete more cases and types of error messages than level 1.

search-period
This gives the number of seconds to watch a subscribers address after receiving a bounced message. If a second bounce is received before this time period is up, the offending user will be removed from the list. Hence, a longer search period will be more likely to remove users from the list.

Example:
The following line uses the maximum level of error analysis. If two errors are received within 7 days of one another, the subscriber will be removed from the list.

```
error_analysis 9 604800
```

Note:
If no **error_analysis** line appears in the config file, error analysis will be disabled for all lists.

default_arch_dir ***directory***

This instruction specifies the default directory for archived files. If the config file directive is not present, this defaults to the following:

```
$LPDIR/archives/list-alias
```

ListProc must have write permissions in the archive directory. ListProc will always maintain its indices under $LPDIR/archives, regardless of where the actual logs reside. You must have this directory on your system.

The site manager can override this for individual lists by explicitly setting the archive directory for the list, either with the **CONFIG** command or the **farch** utility. (See Section 0, Configuring a Mailing List, on page 41.)

5.2. Comments and Continued Lines

The pound sign (#) can be used anywhere within the config file to comment out the rest of that line. Logical config file lines that are too long to fit the width of your text editor can be broken on to multiple lines by escaping the new line character with a back slash (\) at the end of the line.

6. Site Management Issues

6.1. Management Tips

One of the keys to successful management of a ListProc server is understanding the types of problems that are likely to arise, and trying to catch them before they ???? This section analyzes some of the more common problems, and suggests ways of dealing with them.

Coping With Mail Loops

Email loops generally occur when two automatic response systems send email back and forth between one another. For example, suppose George Finkelmeyer, a subscriber on your "Far Side" list, goes on vacation, and leaves a vacation program that responds to all incoming mail with a message such as this:

```
I am currently on vacation, but I will no doubt thoroughly
enjoy reading your mail upon my return.

      Sincerely,

            George Finkelmeyer
```

When this message is received by the list, it will be transmitted out to all of the subscribers, including George Finkelmeyer. This will generate another vacation message, which will again be sent out to the list, *ad infinitum.* If the loop were to go unchecked, it would generate a tremendous amount of traffic to the list, causing congestion on your local network, and bogging down your server. Additionally, the accumulated messages could easily fill up your hard drive, and potentially bring down your entire system.

Fortunately, ListProc has a number of built in features that stop such loops. For example, in this case, ListProc would notice a duplicate checksum the second time the message reached the system, and it would not propagate it any further.

Setting a Daily Message Limit

In spite of ListProc's loop detection capabilities, there is always the chance that some loop will go undetected. **As a last ditch measure against the propagation of email loops, you should set a MESSAGE-LIMIT for all lists on your system.** When a list's MESSAGE-LIMIT is set, ListProc will keep track of the total number of messages (both regular list messages and error mail) that come in for the list in one day. Once the message limit is reached, ListProc notifies the list owner and stops processing additional messages until it is explicitly reset. Additional messages are queued for processing sometime later, after the list owner has had a chance to examine the problem, and reset the list with the FREE command. The message limit is also automatically reset the next

day. For more information, see the discussion of the MESSAGE-LIMIT option on page 51.

Error Mail

When managing large lists over a distributed network such as the Internet, bounced mail, bad network connections and other problems are unavoidable. Due to ListProc's separation of management tasks, bounced messages and errors for a particular list are passed off to the owners of that list (or to ListProc itself, if AUTO-DELETE-SUBSCRIBERS has been set for that list). Occasionally there will be problems with outgoing server messages (responses to SUB commands, and the like), but this usually doesn't amount to very many messages.

The main occasion in which the site manager must intervene is if the **system** mail method is being used, and ListProc's SMTP connection to your mail host (as defined by the **mta_host** config file directive) fails for some reason. In this case, ListProc queues the message in the **LPDIR/mqueue** directory and notifies the system manager of the error. The following SMTP errors trigger this behavior:

```
421   service unavailable
451   host aborted
452   disk full
500   command not recognized
501   syntax error
502   command not implemented
503   bad sequence of commands
504   parameter not implemented
552   too many recipients
554   transaction failed

SMTP connection time out
```

In most cases, the messages queued in the **LPDIR/mqueue** directory can be resent using the **pqueue** command. Alternatively, you can simply start up the **queued** program to check this directory and periodically attempt to re-send the mail. (See the **LPDIR/doc/queue.1** man page for information on running **pqueue** and **queued**.) Occasionally, however, errors in the queued file (such as an empty subscriber address) will prevent the message from going out properly. If this happens, you can simply edit the queued file to remove the bad commands, and try sending it again. If there is only one recipient left on the message, and the address is clearly invalid, you can simply delete the queued file.

6.2. Ignore

The system's home directory each list directory contains a **.ignored** file, containing the address of users whose requests and list messages should be discarded. The default **LPDIR/.ignored** file contains entries for server, bin and sys. You may also want to add entries for each of the lists defined on your system, to make sure you don't somehow end up with lists looping with the ListProc server account, for example. A list's **.ignored**

file also contains an entry of its alias and full e-mail address as well as the server account's full e-mail address.

Entries in this file may also be regular expressions as shown below.

Ignore Command
ignore *list password address*

Alias an existing subscriber to a new address, or ignore postings from the specified address. Both requests are synonyms of the more generic *put* request (put <list> <password> alias <address> <address> and put <list> <password> ignore <address>).

<address> and <new-address> can be regular expressions.

Examples:

```
alias ermis ermis1 uucp!bar.com!foo foo@bar.com
alias ermis ermis1 ^@.*:(.*)@(.*\..*) \1@\2
```

The latter will remove source routing from all addresses and convert them to the user@domain syntax; for example:

```
@GATEWAY:USER@HOST.DOMAIN
```

will be converted to USER@HOST.DOMAIN

```
alias ermis ermis1 [^!@]*!([^!@.]*)!([^!@]*)@.* \2@\1.uucp
```

This will convert GATE!HOP!USER@UUCP.SOME.COM to USER@HOP.UUCP

```
ignore ermis ermis1 foo@bar.com
ignore ermis ermis1 ~<.*\.com|.*\.edu>
```

The latter restricts postings to .com and .edu domains only.

For additional information on regular expressions, you can FTP zoo.toronto.edu.

6.3. Alias

It is possible that a peer/subscriber/newsgroup's/gateway's mail may arrive differently than subscribed, which may raise subscription issues and prevent the person from posting or changing subscription options.

For example, when using the conversion script provided to change your lists from VM Listserv to ListProc, BITNET only addresses are rewritten as follows:

```
user%node.bitnet@interbit.cren.net
```

This is fine, except that when that user sends mail to the list or ListProc, there is no way to predetermine which gateway the mail will traverse. In the end, the address, although in the form above in the subscribers file, may appear to the ListProc as:

```
user%node.bitnet@pucc.princeton.edu
```

To get the ListProcessor to accept that address as equivalent to what is in the subscriber's file, the following alias will work:

```
^(.+)%(.+)\.bitnet@.+ \1%\2.bitnet@interbit.cren.net
```

This alias should go in the top level **.aliases** file if you convert VM Listserv lists to ListProc and leave the pre-assigned gateway as interbit.cren.net. If you change the gateway, change it in the alias or add it, since aliases can be added.

Another useful top level alias deals with organizations that have clusters of many machines. When mailing into the same domain, some fully qualify and others do not. The following alias will accept all non-fully qualified e-mail address as the equivalent cren.net address:

```
^(.+)@([^.])$ \1@\2.cren.net
```

In the case above, the ([^.]) accepts anything but the period, so that user@foo.bar is not converted to user@foo.bar.cren.net.

The *alias* command allows the site manager and list owners to add new aliases to the **.alias** file

For each new list, the system adds the following alias:

```
^@.*:(.*)@(.*) \1@\2
```

This removes source routing.

Warning: This command can have disastrous results if it is set up improperly. Be sure to test and verify your information when using *alias*.

7. List Management

The configuration command described in the previous section provides many of the tools necessary to maintain a mailing list. A number of other tools allow list owners to maintain aspects of the list not directly related to the list configuration. This includes adding and removing subscribers, changing settings for specific subscribers, adding and removing files from the list archives, and editing list subscribers, aliases, and ignore files.

Note
All references to passwords in this section refer to the *list* password, rather than site manager or end user passwords. In all cases, however, it is possible for the manager to issue these commands and use the system password instead.

!!Caution!!
Commands sent to ListProc are limited to 1024 characters in length. (Since most email agents limit lines to about 80 characters in length, this translates to about 13 lines in an email message.) For this reason, complicated operations should be split into multiple commands.

7.1. Defining a List

Both the site manager and the list owner have responsibilities in defining lists. The site manager is the only person who can create a mailing list. Additionally, root access is required because you must add system wide mail aliases for both the server and list to receive mail from the network. Once a list has been created, the both the site manager and the list owners can modify the list configuration to suit their needs.

System Mail Aliases

Each list on your server requires that three aliases be added to the system's aliases file. (This is usually /etc/aliases.) These aliases are piped to the *catmail* utility included with the distribution. Catmail appends the messages to the appropriate file, depending on its command line arguments (see appendix C for a detailed description of catmail).

Each list should have the following three aliases:

```
listname: "|/LPDIR/catmail -L LIST-NAME -f"
owner-listname: "|/LPDIR/catmail -L LIST-NAME -f -e"
listname-request: "|/LPDIR/catmail -L LIST-NAME -f -o"
```

LPDIR should be replaced with the full path to server's home directory, where the binaries exist. For example, if ListProc is installed in the /home/server directory, the aliases for the "foolist" list would be as follows:

```
foolist: "|/home/server/catmail -L FOOLIST -f"
owner-foolist: "|/home/server/catmail -L FOOLIST -f -e"
foolist-request: "|/home/server/catmail -L FOOLIST -f -o"
```

Note: AIX users may want to use *smit* to set the mailer's alias database. The location of the mail aliases file and format will vary depending on your O/S and MTA

Creating a Mailing List

The principle tool in defining lists is the *initialize* or *new-list* command. Only the manager can issue this command. You define a new mailing list by specifying its name, full address and configuration options as follows:

```
initialize manager-password list-alias &
      list-address configuration-options

OR

new-list manager-password list-alias &
      list-address configuration-options
```

Initialize uses the same options and syntax for options as does the *configuration* command described in the next section. You must define at least one configuration option for a mailing list to be created. At a minimum, you should define some of the owners of the list, as ListProc's behavior is undefined for lists without owners.

Example:
The following two commands show correct usage of the command:

```
init mypasswd spam spam@python.com archive, &
      owners foobar, archive, hidden, owner-subscriptions,
&
      auto-delete, password test1, digest daily 12:00, &
      reply-to-list-always, message-limit 50
```

The minimum required to create a list is as follows:

```
init mypasswd spam spam@python.com owners eidle
```

The next example is **incorrect**, since it does not include any configuration options.

```
init mypasswd spam spam@python.com
```

The following example is syntactically correct, but a bad idea, as it doesn't define any list owners:

```
init mypasswd spam spam@python.com REPLY-T0-LIST
```

ListProc uses the following defaults in the absence of the corresponding configuration options:

- Owner controlled
- No archives
- Headers are handled using the *reflector* option
- Not hidden
- Anyone may post to the list
- Anyone may subscribe to the list
- No digests

- No message limit
- List password is the list name, in uppercase

Configuring a Mailing List

List owners can change list parameters except those marked with an asterisk (*) at any time without restarting the server daemon. This provides owners with a wide range of options and flexibility. If an owner sends a configuration command with one or more incorrect parameters, all valid configuration parameters will be applied and the lists owner(s) will be notified.

Modifications to the lists configuration are made through the configuration command. (Abbreviated "config") The syntax for the configuration command is as follows:

```
configuration list-name password [option [args]] &
      [, option [args]Ö]
```

Note that the option parameters are not required. If only the list name and password are specified, the command returns the lists current settings.

The following examples may help you get a feel for how the config command might be used in an email to your ListProc server. Full descriptions of each of the available options are included below. Note that the ampersand characters (&) are used to continue the command onto the next line.

Owner example:

```
configuration test pass1 archive - messages, &
  archives-to-owners, review-by-all, auto-delete, &
  owner-subscriptions, comment Test List, &
  default mail ack conceal no password XyX &
  preference CCSUBSCRIBE preference CCUNSUBSCRIBE, &
  set-disable mail digest conceal yes password, &
  delivery-errors-to root@your-site server@your-site, &
  digest daily 15:30, message-limit 100, &
  moderated-edit you@your-site, owners foo@bar.com, &
  subscr foo@your-site, reply-to-list-always
```

Manager example:

```
conf test pass1 archive /tmp - - - digests, &
  compressed-archives, message-limit 100
```

Options are a comma-separated list of one or more of the keywords listed below. Options marked with "m" can be set only by the site manager.

(m) ARCHIVE [*dir*] [*filespec*] [*arch-name*] [*password*] &
[messages|digests]

This turns on archiving for the list. All arguments are optional, as ListProc includes defaults for each of these values. If you do not wish to set a particular value, but you *do* want to set some of the values after it, use a minus sign (-) as a place holder in your

command. This will allow ListProc to correctly parse the command, even when some arguments are unspecified.

dir
Specifies the destination directory for list archives. The default directory for list archives is specified by the **default_arch_dir** directive in ListProc's config file.

arch-name
The name of the archive.

password
A password to use when accessing the archives. This password is completely independent of the list password, and of individual subscribers' passwords. If a password is specified, users must provide it in order to get archive files.

messages|digests
These specify whether ListProc should archive list messages, or digests of the list messages. If "**messages**" is chosen, the archives will consist of accumulated messages, in standard UNIX mbox format.

The period of the archives is determined by the filespec (see below). If the list is set to archive digests, the filespec is *not* used to determine the period of time, since digests are simply added to the archive when they are complete.

filespec
This defines how the names of the archive files should be constructed. Filespecs can contain any lower case characters of the alphabet, numerals, the underscore and one or more of the following special character sequences (more detailed descriptions of these options are below):

%a	contents of Archive-Name header line
%c	the current message count
%d	day of month (01 - 31)
%h	month name (Jan - Dec)
%j	Julian date (001 - 366)
%m	month (01 - 12)
%n	issue number (extracted from Volume # Number # line)
%v	volume number (extracted from Volume # Number # line)
%w	week number of month (a - f)
%y	year (00 - 99)
%1	first word of first non-blank line of file
%#	digest number (if archiving digests)
%%	the character %

When **%a** is included in the filespec, ListProc expects to find the **Archive-Name:** header line in either the header or the body of all messages sent to the list. (This header line specifies the archive name to store to. This is how various "source" news groups archive messages.) Messages that do not contain the **Archive-Name:** line will not be archived. Note that **%a** cannot be used when archiving digests, as each message in the digest might have a different **Archive-Name:** line.

%v and **%n** are usually specified together and expect the first line of the body of the message to be something like this:

```
Volume 3 Number 45
```

This is useful when archiving an electronic newsletter or magazine.

The system default filespec is **log%y%m**.

Example:

```
arc /ftp/list-archives/test log%y%m - - messages
```

This would store all list archives in message format in a directory called /ftp/list-archives/test. The archive files would be named according to the month and year. If it were December 1994, the file would be named log9412.

Example:

```
arc /ftp/list-archives/misc %a
```

This will store archives only if the **Archive-Name:** appears as a header or on a separate line in the body of the message with the specified file name. For example, if an incoming mail message contained the header line

```
archive-name: census-data-1994
```

it would be archived to the file /ftp/list-archives/misc/census-data-1994.

ARCHIVE [*password*] [messages|digests]

This is the list owner version of the archive command. Note that not all options are available to list owners.

NO-ARCHIVE

Turn off archiving of messages or digests.

(m) COMPRESSED-ARCHIVES

Keep archives in compressed format.

(m) UNCOMPRESSED-ARCHIVES

Keep logs in uncompressed format.

ARCHIVES-TO-ALL

Make list archives and archive indices available to anyone who requests them.

ARCHIVES-TO-OWNERS

Only owners have access to the list's archives and archive indices.

ARCHIVES-TO-SUBSCRIBERS

Only owners and subscribers have access to the list's archives and archive indices.

AUTO-DELETE-SUBSCRIBERS

Turn on auto-deletion of subscribers whose addresses cause delivery problems.

When outgoing list mail bounces, ListProc will attempt to determine the cause and severity of the error. If ListProc determines that the error is unrecoverable (i.e. No Such Local User), it will auto-delete the subscriber. If ListProc detects that the error is transitory (i.e. Disk Quota Exceeded), ListProc will set the user's mail option to *postpone*. In either case, the list owner will be notified of the action taken.

ListProc's auto deletion features can be fine tuned using the **error_analysis** config file directive. (See page 33.) If no **error_analysis** line is in the config file, error analysis is not performed, and this option has no effect.

NO-AUTO-DELETE-SUBSCRIBER

Turn off auto-deletion. This is the default behavior for new lists.

PUBLISHED-LIST

If a list is PUBLISHED, then it will be made known to the *'global-update-server'* specified in the ListProc config file. The global update server then automatically adds the list name and description line to its own remote.lists file. This will allow users on other hosts to search for it and subscribe to it without knowing it's location.

UNPUBLISHED-LIST

This will make the list know only locally. This is the default state.

CLOSED-SUBSCRIPTIONS

Do not accept any more subscriptions. This is different from owner subscriptions (see below) because this command will simply not accept any subscription requests. The person requesting subscription will be notified that no subscriptions are being accepted for the list.

OWNER-SUBSCRIPTIONS

Subscriptions are monitored by the owners or the subscription managers who get a chance to approve them or not. (see below).

OPEN-SUBSCRIPTIONS

This option is reverses the effects of CLOSED-SUBSCRIPTIONS and OWNER-SUBSCRIPTIONS, restoring the subscription mechanisms to their normal operation.

WIDE-OPEN-LIST

Open up subscriptions to anyone, make the list visible when users issue a list command, and allow anyone to review the list, get statistics on usage or access to the list's archives. This option also allows anyone (whether subscribed or not) to post to the list.

A wide open list is equivalent to:

visible
send-by-all
open-subscriptions
archives-to-all
review-by-all
reflector.

COMMENT *string*

Specify the one-line content of **To:** header line on outgoing list mail. For example, if you issued the command:

```
config happy-list mypasswd comment Happy Joy Mailing List
```

The **To:** line on mail sent out from the list "happy-list" would look like this:

```
To: Happy Joy Mailing List <happy-list@joy.com>
```

To have the ListProc an empty comment, use this command:

```
config happy-list password comment ""
```

This would produce the following **To:** line:

```
To:  <happy-list@joy.com>
```

Note that this is *not* the same as specifying NO-COMMENT.

Note:
The following characters are not allowed in the comment field:

```
[]^<>‘*?,| and the newline characte
```

Additionally, since the comment string is included directly into the **To:** header of outgoing messages, you should escape the quotation, colon and apostrophe characters (" : ') with a backslash (\) to make sure your MTA and other people’s mail readers deal with it appropriately.

NO-COMMENT

This removes the definition of the comment line from the list’s config file, and instructs ListProc to insert “Multiple Recipients of List” in the **To:** line. In the example above, the command

```
config happy-list password no-comment
```

would produce the following **To:** line:

```
To: Multiple Recipients of List <happy-list@joy.com>
```

Note that this is *not* the same as defining an empty comment string. (See above)

This is the default for a new list.

DEFAULT *mailmode value* [*mailmode value*]Ö

This allows you to set default behavior for a number of aspects of your list. Each of the ***mailmode*** values can be set to ListProc's default by using the empty string "" or " as the ***value*** in the command. The system defaults are as follows:

address fixed
mail ack
system generated random password (as opposed to a default password)
conceal no

Acceptable ***mailmode value*** pairs are described below.

address variable|fixed
This tells ListProc whether or not to allow subscribers to change the address in the list. If this is set to **fixed**, they will not be able to change their address. If it is set to **variable**, they will be able to change their address with the **set** command:

```
set list-name address current-password new-address
```

(See the ListProc User Manual for more detail on the **set** command.)

mail ack|noack|digest|postopone
This sets the default behavior of outgoing mail to subscribers.

ack	A list subscriber will receive a copy of his/her posting to the list.
noack	A list subscriber will not receive a copy of his/her posting.
digest	The subscriber will only receive the digest, not individual postings.
postpone	The subscriber will not receive any postings or digest.

Individual subscribers can change their **mail** option with the **set** command. For example, the following sets mail to digest:

```
set list-name mail digest
```

password *default-password*
This option sets a default user password upon initial subscription. If not set, a random system generated password is assigned to each user. The need for a password arises from the interactive use of ListProc. The password is the only means to authenticate a user. If a default password had been set previously, you can reset ListProc to generate random passwords for subscribers by setting an empty default password:

```
conf list-name list-password default password ""
```

The quotation mark, comma, and apostrophe (" , ') cannot be used in valid passwords.

conceal yes|no
This determines whether or not the user's name and e-mail address will be shown during a **review** command.

preference *CC-option*
The **preference** field defines the default preference value for all list owners added after the **config** default command is given. (The initial list owner is given the default preference of CCERRORS.)

Individual list owners can change their preference value using the **set** command. More than one CC-option can be set for each owner, and once set, CC-options can be un-set by adding a minus sign to the beginning of the option name when preferences are set. For example, to un-set the CCERRORS preference, an owner could send this command:

```
set list-name preference -CCERRORS
```

Owners can view their current preferences (and other settings) by sending the **set** command with no arguments:

```
set list-name
```

The following CC-options are available for **list owners**:

CCERRORS
The list owner will be copied on every listproc error message generated.

CCSET
The list owner will be copied on every set command that a subscriber issues (i.e. when a subscriber sends changing mail options to digest).

CCSUBSCRIBE
The list owner gets copied on all subscription requests.

CCRECIPIENTS
The list owner gets copied on all get requests for archives.

CCINFORMATION
The list owner gets copied on all info requests.

CCSTATISTICS
The list owner get copied on all statistics requests.

CCPRIVATE
If the list is set up so that only subscribers can post, this CC-option will cause the list owner to receive a copy of all posts attempted by non-subscribers.

CCREVIEW
The list owner(s) gets copied on all review requests.

The following are valid for ListProc **site managers**:

CCRUN
The manager gets copied on all run requests.

CCIGNORE
The manager is notified when a message is ignored for whatever reason.

CCGET
Copies of all get requests are sent to the manager.

CCINDEX
Copies of all index requests are sent to the manager.

CCLISTS
Copies of all list request are sent to the manager.

CCRELEASE
Copies all release requests.

CCHELP
Copies of all help requests are sent to the manager.

The final two options are available to **both the site managers and list owners** alike:

CCALL
The list owner(s) and/or ListProc site manager get copied on all of the above which are applicable. Be cautioned that can generate a lot of mail. No corresponding -CCALL is available!

CCDUP
This will prevent owners and managers (depending on who has it set) from getting duplicate messages. One copy of the error message is sent and the remainder are removed by ListProc.

Example:
The following command sets the default owner and subscriber settings for the "foolist" list:

```
config foolist foopass default  mail digest &
      password sub_pass  preference CCALL &
      preference -CCSUBSCRIBE
```

New subscribers will automatically have their mail set to "digest" and their password set to "sub_pass". New owners will be copied on all messages except SUBSCRIBE messages. The settings of existing subscribers and owners will not be affected.

DELIVERY-ERRORS-TO *address* [*address*]Ö

This specifies the recipients of delivery error messages. (Delivery error messages are mailer error messages, as opposed to error messages generated by ListProc.) The address(es) specified by this option will be given the power to remove users with the DELETE command with the list's password. (See the ListProc Owners manual for more information on the DELETE command.)

If no DELIVERY-ERRORS-TO recipients are specified, delivery errors will go to all of the owners.

REMOVE-ERRORS-TO *address* [*address*]Ö

This removes addresses from the list of DELIVERY-ERRORS-TO recipients.

DIGEST *frequency when* [*maxlines maxbytes*]

This enables ListProc's digest features for the given list, and specifies when and how the digests should be created.

frequency when
This parameter defines how often ListProc should generate digests, and allows the sender of the command to optionally define when these digests should be distributed. Valid *frequency*, *when* pairs are as follows:

```
daily hh:mm
weekly Sunday|Monday|Tuesday|...|Saturday
monthly
```

The delivery time for daily digests is specified in 24 hour or military time. Weekly digests are distributed at midnight on the day specified. Monthly digests are always distributed at the end of the last day of the month. Thus, specifying a value for ***when*** for these will produce an error.

Example
To set the "jabralta" list to deliver digests at 4:00 p.m. daily, you would issue the following command:

```
config jabralta jab-passwd digest daily 16:00
```

maxlines maxbytes
If you specify the number of lines or size in bytes (or both), digests will be distributed as soon as these limits are exceeded. Thus it is possible to distribute multiple digests within the time period specified by the ***frequency*** parameter.

In order to correctly distinguish these parameters from the optional ***time*** parameter when the ***frequency*** is daily or weekly, the ***time*** parameter must be present when specifying values for ***maxlines*** and ***maxbytes***. If you do not wish to specify a time, a minus sign (-) can be used as a place holder. Specifying a zero for either ***maxlines*** or ***maxbytes*** makes that quantity unlimited.

Example
To send out digests when 50,000 lines are accumulated, the following would do the trick:

```
config list-name password digest monthly 5000 0
```

Since the digest ***frequency*** is set to "monthly", digests will be sent out *at least* every month. If the list is active enough to accumulate more than 5000 lines of traffic before the month is over, a digest will be sent out when this occurs. Digest distribution is not triggered by the total number of bytes in the digest, since ***maxbytes*** is zero.

NO-DIGESTS
Turn off collections of digests.

DISABLE *request* [*request*]Ö
Disable use of the specified request(s). For example, to disable the ability of anyone to review the list, the following command would be sent:

```
conf list-name password disable review
```

ENABLE *request* [*request*]...
Enable use of the specified request(s).

SET-DISABLE *mode* [*value*] [*mode* [*value*]]...
This prevents users from setting the specified ***value*** of the user mode ***mode***. Hence, the users will not be able to change their subscription options from the list default in these cases. Valid ***mode***, ***value*** pairs are exactly those available to users through the SET command:

```
mail ack|noack|postpone|digest
conceal yes|no
password
```

Notes:
It is not necessary to specify a ***value*** when disabling password changes. SET-DISABLE cannot be used to disable owner preference options.

Example
If the list owner wanted to maintain a policy of no hidden users, the following configuration command could be used to prevent users from concealing their subscription:

```
conf list-name list-pass set-disable conceal yes
```

SET-ENABLE *mode* [*value*] [*mode* [*value*]]Ö

This simply re-enables SET commands disabled by the SET-DISABLE configuration command. The syntax is the same as in the case of SET-DISABLE.

Example
To reset the ability of users to conceal themselves, send the following command:

```
conf list-name password set-enable conceal yes
```

FORWARD-REJECTS

If the list is set to **FORWARD-REJECTS**, error messages generated by incorrect subscriber requests are sent to the list owner, rather than back to the sender.

DONT-FORWARD-REJECTS

Forward all ListProc-generated error messages to the senders. This is the default.

KEEP-RESENT-LINES

Keep **Resent:** header lines. Note that if the config file **header** directive specifies that the Resent lines should NOT be preserved, the global config file takes precedence.

DONT-KEEP-RESENT-LINES

Don't keep **Resent:** header lines. If the global config file contains a header line that specifies that the **Resent:** lines should be kept, the config file takes precedence.

HIDDEN-LIST

Hide the list from "lists" and "index" requests. ListProc makes certain assumptions about hidden lists and sets the following other options automatically. It restricts access to archives, review and statistics to subscribers and owners only. If you want your hidden list to exhibit other characteristics, you must explicitly set the corresponding options.

VISIBLE-LIST

Turn on visibility for lists and index requests, but do not affect the rest of the options.

(m) MANAGER-CONTROLLED

Only the manager can remove owners and change the list password. This lets site managers temporarily take control of a list while changes are being made in ownership or policy, or to enforce a policy that the owner be known and a member of the institution. When a list is manager controlled, an owner can't change the owner, archive, or password options.

(m) OWNER-CONTROLLED

Allow list owners to change the owner, archive, and password options. This is the default.

MAX-MESSAGES-PER-DAY *number*
MESSAGE-LIMIT *number*

Specify the maximum number of messages to be processed in one day. Once this limit has been reached, message processing stops, and the list owners are notified. (Messages are still queued, but no further processing takes place.) List owners can restart the processing of list mail with the FREE configuration option.

We recommend setting the message limit as a last defense against any mail loops that ListProc does not catch. If you have a busy list, set this to a high value.

NO-MESSAGE-LIMIT

This turns off the restriction on the number of messages as specified by MAX-MESSAGES-PER-DAY or MESSAGE-LIMIT. This is the default for new lists.

MODERATED-EDIT [*address*]Ö

This enables moderation of the list, and allows the selected moderators to edit messages before they are sent to the list. If no moderator addresses are specified, incoming messages will be sent to the list owners for approval.

MODERATED-NO-EDIT [*address*]Ö

This enables moderation, but does not allow the moderators to edit the incoming messages. If no moderator addresses are specified, incoming messages are sent to the list owners for approval.

UNMODERATED

No postings are moderated. This is the default for new lists.

REMOVE-MODERATORS *address* [*address*]

Remove addresses from the current list of moderators. If the last moderator address is removed, incoming list messages will be sent to the owners for approval. Each new address is separated by a space.

OWNERS address [address]Ö

Specify the email address(es) of new owners of the list. Each new address is separated by a space.

REMOVE-OWNERS *address* [*address*]Ö

Remove the specified owners. Each new address is separated by a space.

PASSWORD *string*

Set the list's password to ***string***.

SUBSCRIPTION-MANAGERS *address* [*address*]Ö

Specify who will approve subscriptions if the list is set to OWNER-SUBSCRIPTIONS. (If no subscription managers are specified, subscription requests are sent to the list owners for approval.)

Subscription managers are allowed to use the following commands to add and remove users from the list:

```
add
delete
system ... subscribe
system ... unsubscribe
review
statistics
put (for subscribers and aliases)
```

Additionally, when ARCHIVES-TO-OWNERS is on, the subscription managers have the same priviliges as the owners for the following requests:

```
index
get
search
```

For more information on these commands, please see the ListProc Owners Manual.

REMOVE-SUBSCRIPTION-MANAGERS *address* [*address*]Ö

Remove an address from the list of subscription managers.

REMOVE-ALL-SUBSCRIPTION-MANAGERS

This is a shortcut for the previous command. It removes all subscription managers.

REFLECTOR

This option instructs ListProc to make headers on postings to the list appear as though they were processed by a sendmail reflector. It will pass **To:** and **CC:** as both received, and no comment, and will exclude a pre-set **Reply to:** (i.e. You cannot have *reflector* and *reply-to-list*). This is the default and must be explicitly turned off when creating a new list.

NO-REFLECTOR

Turns off the reflector option.

REPLY-TO-LIST

If no **Reply-To:** header is present on an incoming list message, a **Reply-To:** header will generated that specifies the list's address. **Reply-To:** headers already present on incoming messages are preserved.

REPLY-TO-LIST-ALWAYS

Set the **Reply-To:** header line on outgoing list mail to the list's address, regardless of the presence or absence of a **Reply-To:** in the original message.

REPLY-TO-SENDER

If no **Reply-To:** header is present on an incoming message, generate a **Reply-To:** that points back to the address in the **From:** header of the message. If a **Reply-To:** already exists, leave it alone.

REPLY-TO-SENDER-ALWAYS

Set the **Reply-To:** header line on outgoing messages to the **From:** address regardless of whether or not the original message contained a **Reply-To:** header.

REPLY-TO-OMITTED

With this option set, ListProc will not generate **Reply-To:** headers. Existing **Reply-To:** headers on incoming messages are preserved. This is the default setting.

REVIEW-TO-ALL
REVIEW-BY-ALL

Anyone has permission to review the list. This is the default.

REVIEW-TO-OWNERS
REVIEW-BY-OWNERS

This setting restricts REVIEW commands on this list to owners only.

REVIEW-TO-SUBSCRIBERS
REVIEW-BY-SUBSCRIBERS

This sets the ListProc to accept REVIEW requests for this list only if they are issued by subscribers or the lists owner(s).

STATISTICS-TO-ALL
STATISTICS-BY-ALL
STATS-TO-ALL
STATS-BY-ALL

ListProc will accept statistics requests for this list from anyone. This is the default behavior.

STATISTICS-TO-OWNERS
STATISTICS-BY-OWNERS
STATS-TO-OWNERS
STATS-BY-OWNERS

Restrict statistics to owners only.

STATISTICS-TO-SUBSCRIBERS
STATISTICS-BY-SUBSCRIBERS
STATS-TO-SUBSCRIBERS
STATS-BY-SUBSCRIBERS

Restrict statistics to subscribers and owners only.

SEND-BY-ALL

Open up postings to everyone, including non-subscribers. This is the default behavior.

SEND-BY-OWNERS

Restrict postings to owners only (one-way list). This does not imply a moderated list.

SEND-BY-OWNERS-CONFIRM

Restrict postings to owners only. In addition, the owner must place the following line in the body of the message:

```
confirm: list-password
```

where *list*-password is the manager's password for this list. ListProc will remove this line prior to distributing mail.

SEND-BY-SUBSCRIBERS

Restrict postings to subscribers and owners only.

SEND-BY-SUBSCRIBERS-CONFIRM

Restrict postings to subscribers and owners only. Additionally, the following line is required in the body of the message:

```
confirm: subscribers-password
```

where *subscribers-password* is the user's password for this list. ListProc will remove this line prior to distributing the messages.

(m) THREADS *number*

The THREADS option specifies the maximum number of threads that can be allocated to the processing of this list. The default value for this is one. If a larger number is specified, the list process will spawn multiple processes to send out the list's mail. The threads used by the list count toward the total defined in the system config file. This means that your list is not guaranteed to actually get ***number*** threads if other lists have used them first.

7.2. Email list Management Commands

[quiet] add *list password address name*
[quiet] add *list password* {*address name*} {*address name*}Ö

The **add** command allows list owners to add subscribers to the list. If you wish to add more than one user per command, you need to enclose the information for each new subscriber in braces, as shown above. ListProc defaults to sending out the standard new user welcome message to new subscribers. If the **quiet** key word is prepended to the command, this message will not be sent.

[quiet] delete *list password address* [*address*]Ö

The **delete** command allows the owner to remove subscribers from the list. Note that the subscribers to be removed are specified by address only, rather than by address and name, as in the **add** command. By default, the unsubscribed users will receive the typical signoff message for this list. If **quiet** is used, this message will not be sent.

approve *list password tag* [*tag*]Ö

Approve the messages corresponding to the specified tag numbers. This tells ListProc to send them to the list. This command applies only to lists which are configured as MODERATED-NO-EDIT.

discard *list password tag* [*tag*]Ö

Discard the messages identified by the specified tag numbers. These messages will *not* be sent to the list. This applies only to lists which have been configured as MODERATED-NO-EDIT.

edit *list password file-description* [-nolock]

Obtain the specified file for editing. By default, ListProc will lock the specified file, to prevent requests from users or other managers from changing the file while the owner is

editing it. Requests that would normally modify the file will be processed only after either the file is put back (with the **put** command) or the list is unlocked (with the **unlock** command.)

file-description
This is the file that the user wishes to edit. Valid values for ***file-description*** are listed below, along with a brief explanation of the purpose of the file. More detailed descriptions of these files can be found in Appendix E.

```
subscribers (subscriber addresses and settings)
aliases     (table of address translations)
news        (information about peered newsgroups)
peers       (information about peered lists)
ignored     (users whose messages should be ignored)
info        (information about the list)
welcome     (welcome message for new subscribers)
```

-nolock
The **-nolock** parameter disables ListProc's normal locking of this file. This can be useful for just viewing the file. It is generally a bad idea to put a file back that was edited but not locked, as any changes that others have made in the interim will be lost.

Examples
To retrieve and edit the info file for the smurdgy list, use the following command:

```
edit smurdgy smurdgy-pass info
```

To view the subscribers file, you could issue this command:

```
edit smurdgy smurdgy-pass subscribers -nolock
```

put *list password file-description* [*args*]

This allows the server manager and list owners to modify a list's files. In particular, files that have been retrieved with the **edit** command can be put back on the server with **put**.

file-description
This specifies the file to be modified. Allowable options for this are the same as those for the **edit** command above.

args
When ***file-description*** is "aliases" or "ignored," additional arguments can be given to add a new item, rather than replacing the entire file. (Additional arguments are supported for these two files only - other files must be completely replaced.) Additional arguments for the aliases file consist of two addresses: the original address, and what it is to be translated to. With the ignored file, the additional argument is treated as an address to be ignored.

Note: the putting of additional aliases and ignored addresses can be done more easily with the **alias** and **ignore** commands respectively.

When optional arguments are not specified, the existing file is replaced by the lines following the **put** command.

Example: adding a new alias
Use the following command to add an alias that translates "mary@test.wilder.com" to "mary@wilder.com":

```
put listname listpass aliases mary@test.wilder.com &
  mary@wilder.com
```

Example : updating the list info file
To put a new information file for the list, use the following command:

```
put foolist foolpass info
The FooList is an interest group for clowns
and jesters worldwide.  List subscriptions
are restricted to members of the international
trickster's union.
```

The text lines after the **put** command become the new info file for foolist.

Caution: When using the **put** command over email, make sure you don't accidentally include your .signature in the message to the server. The inclusion of this extra text in the aliases, ignored, or subscribers files could cause undesirable behavior when ListProc next processes mail.

hold *list password*

This halts the delivery of list mail. Messages sent to the list while it is held are queued for latter processing. Note that unlike the **lock** command, this does not effect the processing of incoming server commands that deal with this list.

free *list password*

Resume delivery of a held list. This also resets the current message count to zero. Hence, **free** can be used to cause a list that has surpassed its message limit to continue processing.

Note: if **hold** was issued by the system manager, it can be **free**-ed only by the system manager. List owners will be unable to free the list in this case.

lock *list password*

Suspend execution of list-specific requests and queue them up for later processing. Owners may still issue such requests unless the list is locked by the manager.

unlock *list password*

Resume execution of list-specific requests, including those queued up while the list was locked. All owners may unlock a list, unless it's locked by the manager.

alias *list password expr1 expr2*

This adds a new alias to the list's aliases file. Aliases use regular expressions to translate addresses into a different form. For a more detailed explanation of ListProc aliases, see Chapter 7. ListProc's regular expressions are discussed in Appendix F.

Example:
Addresses ending with cren.net can be modified to remove subdomain information as follows:

```
alias somelist somepass (.*)@(.*)\.cren\.net \1@cren.net
```

This would change mickey@mouse.cren.net to just mickey@cren.net.

ignore *list password expr*

Adds a new regular expression to the list's ignored file. Mail from addresses that match the expressions in the ignored file will be discarded without being sent to the list. For more information on ListProc's ignored addresses, see Chapter 7. For information on regular expressions, please refer to Appendix F.

Example:
To ignore all postings from pesky@anyhost.domain, use the following command:

```
ignore somelist somepass ^pesky@*.domain
```

reports *list password*

Obtain all reports about the specified local list.

[quiet] set *list* [*option* [*args*]] [for *address* [*address*]Ö]

This allows the list owner to set options for specific subscribers. The arguments for the set command are the same as listed previously. If **quiet** is specified, the users will not be notified that their settings have been modified.

system *list password user-address #user-request*

This allows a list owner or the system manager to execute a command as if they were a different user. The parameters prior to the # specify the list and subscription under which to issue the request. The actual request follows the #.

Example
To obtain a list of the lists to which a user is subscribed, use the **system** command to give the **which** command as that user:

```
system foolist foopass krusty@clown.net #which
```

Note that in this example, the name of the list is unimportant, since the "which" command gets settings from all lists.

purge *manager_password address* [*address*]Ö

The **purge** command allows the ListProc manager to remove users from all lists on the local server. Note that the *manager* password must be specified to purge other users.

A version of the **purge** command is also available for subscribers. The syntax for the subscriber command is just "purge *password*", where *password* is a password from any of their accounts on the server. This is described more thoroughly in the ListProc user reference manual.

7.3. Removing a List

The steps involved in removing a list are actually quite simple:

1) Stop ListProc with the "stop" shell command

2) Edit the "config" file, to remove references to the list(s)

3) Edit the "owners" file, to remove the list's owners

Once that is done, ListProc will no longer know anything about the list. To fully clean up your system, you can add the following steps:

4) Remove the LPDIR/lists/LISTNAME directory for the list in question

5) Use the **farch** command to remove the list's archives (if there are any). The command line would look like this:

```
farch -R archive-name
```

6) Edit your mail transfer agent's aliases file (usually /etc/aliases) and remove all of the aliases for the list.

Note
if you plan to recreate the lists with the same names, it is important that you at least rename the old LPDIR/lists/LISTNAME directory, to make sure that ListProc doesn't try to use any of the old configuration files.

8. Archiving Files

In addition to its mail handling capabilities, ListProc 8.0 includes a suite of file archival and distribution features. ListProc archives contain a full set of access controls, including different levels of control for each archive owner, and the option of password authentication for achieve access. Archive owners can control the logical structure of their archives through the use of subarchives, and simple file descriptions. ListProc also includes the ability to automatically deliver modified files to users, or to notify the users when files have changed.

8.1. ListProc Archive Structure

ListProc archives are composed of four things: The archive directory, two data files, INDEX and DIR, and the archived files themselves. The archive directory holds two data files. It may also contain the archived files, although this is not required. The INDEX file contains a list of all of the subarchives of the archive. (The archive itself is listed in the INDEX file, as the zero-th level subarchive.) Finally, DIR gives the locations, sizes, and descriptions of the archived files.

Subarchives

Subarchives allow the site manager to create a rich structure within ListProc's archives.

Private Archives

Private archives require users to specify a password in order to obtain an index and to receive or subscribe to files. To make an archive private, simply append a password (case does not matter) after its full path specification in every **INDEX** file the archive is defined (its own plus all ancestors' — the same password has to be used in all of these files). Then, that password has to be made known to all users who are to be granted access to this archive. Note that all files in the same private archive can be obtained with the same single password.

DIR

The DIR file contains a list of the files available in the archive. Each file is described by a single line. The format is of the file is as follows:

```
file nparts size-list full-path [description [\]]
[additional-description [\]]*
```

file
The name of the archived file.

nparts
The number of parts into which the file has been split. Setting the number of parts to negative 1 (-1) indicates that the file is binary.

size-list
This is a space separated list of the sizes of each of the parts, in order.

full-path
The full path to the file, excluding the file name. For example, if one of your archives contained the file "/etc/hosts," the ***file*** parameter would be "hosts," and the ***full-path*** parameter would be "/etc."

***description* [\]**
[*additional-description* [\]]*
This is the file description that will appear in the output of the **index** command. The description can be extended onto multiple lines by escaping the newline character with a backslash. For multi-line descriptions, make sure that the each line except the last ends with the \ character.

INDEX

The index contains a list of all subarchives including the archive itself. Each archive is on it's own line. The format is as follows:

[%]*subarchive-name subarchive-path* [*subarchive-password*]

subarchive-name
The name of the subarchive. If the archive name begins with "%", the AFD and FUI will be enabled for this archive.

subarchive-path
This is the full path to the subarchive directory

subarchive-password
This specifies the password users must give in order to receive an index, get files, or subscribe to files from this archive. As with ListProc's other passwords, this password is case sensitive. This password must be the same for all in all of the INDEX files in which the archive is listed.

The first line of the INDEX file should describe the archive itself. Second come the lines describing the first level subarchives. Next are the lines for each second level subarchive, starting with the subarchives of the first first-level archive, then the subarchives of the second first-order archive, etc.

No sibling archives (subarchives of the same parent) may have the same name. Archives on different levels of the hierarchy or that are subarchives of different archives can have the same name. These archives are distinguished by the relative path needed to access them.

Restrictions and Warnings

Archive names and input files must use only lower case characters of the alphabet. Capital letters are reserved for ListProc archive information files.

8.2. Manual Creation of Archives

1. Create a directory under LPDIR/archives or under an existing archive's directory (beware of the hierarchy structure), naming the directory after the archive. The archive's name should be unique among its siblings on that level in the hierarchy. (In any case, *mkdir* will not let you specify a name that already exists!!)

2. Edit the master archive's index file, **LPDIR/archives/listproc/INDEX**, and add the new archive along with its path (and optional password — see below) at the proper place in the file, making sure you adhere to the rules about hierarchy outlined above. The first line of every **INDEX** file should be the archive itself. Also edit (add to) every ancestor archive's **INDEX** file, if the new archive is not the default, again making sure you adhere to the same rules.

3. Create a new INDEX file in the new archive's directory and put an entry for itself.

4. Create an empty DIR file in the new archive's directory.

8.3. Email and ILP Archive Management - the *ARCHIVE* Command

The main mechanism through which ListProc archives are controlled is the **archive** command. The syntax and usage of the archive command is slightly different for site managers and archive owners.

Site Manager Command

```
ARCHIVE archive-name manager-pw CREATE|MODIFY  &
 [action [args]]  &
    [,action [args]] ...
ARCHIVE archive-name manager-pw REMOVE
```

manager-pw
The site manager password

archive-name
The name of the archive

CREATE|MODIFY
These can be used interchangeably. In either case, if the archive does not exist, it will be created, and existing archives will be updated.

REMOVE
Delete the archive and remove all of the included files. If the archive includes subarchives, these will be removed as well.

PASSWORD ***archive-pw***
This specifies the password that archive owners must use to modify the archive. This must slot must be used even if the archive is just being modified. If you don't wish to redefine the archive password, you may replace this argument with a dash (-), to indicate that the previous password should be used.

action **[*args*]**
The following actions are available for the manager version of the **archive** command:

DIRECTORY *dir*|-
The ***dir*** parameter defines the directory in which the archive should be created. This directory should be an absolute path name. If a dash is used, the archive will be created in the default directory.

PRIVATE *user-password*
This requires users to give the specified password in order to access the archives.

PUBLIC
Allow users to access archive files without specifying a password. This is the default setting.

COMPRESSED
Archived files will be stored in a compressed format, to reduce disk usage.

UNCOMPRESSED
Archives will not be stored in compressed format. This is the default setting.

SPLIT *size*
Files will be split into chunks of no more than *size* bytes when they are sent to users.

UNSPLIT
Do not split files. This is the default setting.

AFD-FUI
Enable automatic file distribution (AFD) and file update information (FUI) for this archive.

NO-AFD-FUI
Turn off AFD and FUI. This is the default.

OWNER *address* [ADD|DELETE|UPDATE]
This specifies that *address* is the email address of one of the archive owners. This option can be used multiple times to define multiple owners. UPDATE, ADD, and DELETE control whether or not the archive owner can add, remove, or modify archive files, respectively. If UPDATE is specified, the owner will automatically be able to add files as well. If no access privileges are specified, the owner will be allowed to modify archive settings, but not to alter the actual contents of the archive.

REMOVE-OWNER *address* [,*address*] ...
Remove a list of archive owners.

PASSWORD *archive-pw*
Specify the archive password. If the archive's config file does not contain a password line, you will be required to include this in your command.

Example:
The following command creates an archive called "NetMagazine":

```
ARCHIVE NetMagazine site-pw CREATE PASSWORD iluvthenet &
  OWNER writer@magazine.net, &
  OWNER destro@joe.com DELETE, &
  AFD-FUI, &
  DIRECTORY -
```

The archive password will be "iluvthenet," the owners are "writer@magazine.net" and "destro@joe.com," and AFD and FUI will be enabled. Note that owner destro@joe.com has permission only to remove files, but cannot add or update them.

Owner Command

The site manager and list owners can add, update and remove files from archives through email. When adding or updating a file, the message lines following the ARCHIVE command are interpreted as the file to be added to the archive.

The syntax for this version of the ARCHIVE command is as follows:

ARCHIVE *arch-name arch-password* ADD|UPDATE|DELETE *filename* &
**   [*description*]**

arch-name
The name of the archive to which to add the file. For subarchives, this can also be the path to the archive.

arch-password
This is the archive password.

ADD | UPDATE | DELETE
Specify whether to add a new file, update an existing one, or delete a file from the archive.

description
The file description that will be returned with an index of the archive.

Example:
To continue the example above, if writer@magazine.net wanted to add a new file to the NetMagazine archive, (s)he would send the following command to the ListProc server's email address:

```
archive NetMagazine iluvthenet add Volume1Number1 &
    "The first Issue of NetMagazine"
NetMagazine
The anti-wired no glossy picture magazine
Look ma, better than WIRED !
```

This would add the file **volume1number1** to the archive netmagazine. (Note that the file name will be converted to all lower case, to conform with ListProc's archive filename conventions.) The contents of this file would be as follows:

```
NetMagazine
The anti-wired no glossy picture magazine
Look ma, better than WIRED !
```

To retrieve this file, a user would simply issue this ListProc command:

```
get netmagazine volume1number1
```

Notes

The archive command will work both over email and ILP. ILP though is not yet aware of the new archive-owner class. This means that if a user logs into listproc using ILP and gives the archive owner email address and password, listproc will not recognize this user as an archive owner. However, if the archive owner's address is identical to one of the list owners' addresses, (s)he *will* be correctly authenticated, and the command will go through.

The archive command works fully over email.

8.4. Command Line Archive Management - the farch Utility

ListProc's **farch** utility allows the site manager to create and maintain archives and archive files from the UNIX command line. In addition, **farch** has the capability to automatically split files into multiple parts, for easier retrieval by email later. New archives are created simply by adding files to a non-existing archive name. The correct directories and DIR and INDEX files are created automatically.

The proper syntax for **farch** is as follows:

```
farch [options] [file-list]
```

file-list
A list of the files to be acted upon (archived, removed, etc.)

options
A number of options allow the user to control the way the files are archived. These are described in detail in the next section.

Note:
The LPDIR variable must be properly set for *farch* to find existing archives, and to create new archives in the correct place.

Options

The following command line options are recognized:

-s *size*
Specify the maximum size in kilobytes of each of the subparts. By default, farch will split files into subparts of no more than 64K bytes.

-n
Do not split files when archiving them.

-b
Input files are binary; they are uuencoded before being archived.

-B
Input files are binary; they are neither uuencoded nor split.

-d *dir*
Specify the directory into which the output files are to be placed. If an archive name is specified on the command line, this defaults to **`LPDIR/archives/archivename`**. If no archive name is given, this defaults to **`LPDIR/archives/listproc`**. If the directory or any of its parent directories do not exist, they will be created.

-p *password*
Make an archive private, with the access password "*password*". Users will be required to specify the password in order to retrieve files, get an index, and subscribe or unsubscribe to files through AFD and FUI (see below).

-D *description*
Specify a description for the file. To separate the description from the rest of the command, it should probably be enclosed in single quotes. This option does not make much sense when archiving more than one file at the same time.

-t *file*
Input files are tar-ed into a file which is then archived. By default, the tar file will be uuencoded and split up, as with the -b flag for binary files. This behavior can be overridden by using the -B flag. Note that whatever the path to the file may be, it will be moved to the specified directory as noted by the -d flag.

Be careful that the names of the input files only include relative path names, as users may not be able to extract files with absolute path names without root privileges on their system. Also, make sure that the resulting tar file is not one of the input files.

-u
Update previously archived files with a new versions. The previous comment fields are preserved, unless the -D option is used to specify a new file comment.

-r
Remove the specified files from the archive. If no archive is specified, the files are removed from the "listproc" archive. This flag overrides all of the options listed above.

-a *archive | path-to-archive*
Specify the name of the archive into which the files should be stored. If no archive name is given, farch uses the default of "listproc". Archives on different levels of the hierarchy may have the same name, and in this case a path-to-archive may be specified to

distinguish between them. *path-to-archive* has the form archive[/archive[/archive...]]. If any of the archives in the path do not exist, they will be created with the necessary files.

-A *archive | path-to-archive*
This functions the same way as -a, except that the archive is marked ready for AFD and FUI. (See the section on AFD and FUI below). AFD/FUI can be disabled by using the -a flag, and re-enabled by using the -A flag again.

-Z
Turn off file compression.
File names in the DIR file have to be unique and *farch* will not archive duplicate files.

-R
Remove an entire archive and its subarchives

Examples

Archive **`src/data`** and **`src/data2`** under archive **`listproc`** (the default), using a maximum file size of 1K; the output file(s) will be placed in **`/tmp/tmp`**:

```
% farch -s 1 -d /tmp/tmp src/data src/data2
```

Archive **`/etc/hosts`** under archive **`unix`** (that is **`listproc/unix`**). The -n flag is used to avoid writing the hosts file in multiple parts.

```
% farch -n -a unix -d /etc -p private /etc/hosts
```

Archive **`/etc/password`** under archive **`pub/unix`** (that is **`listproc/pub/unix`**):

```
% farch -n -a pub/unix -d /etc /etc/passwd
```

Tar and archive all files in **`/usr/src`** to the default archive and place the tar file under **`/tmp/tmp`**:

```
% farch -t LPDIR/source -d /tmp/tmp /usr/src/*.c
```

Descriptive messages about these files are added manually into the archive's DIR file. However, the -D option can be used to specify a string as follows:

```
% farch -D 'ListProcessor system files' -t ulistproc.tar \
-n -d /usr/server /usr/server/*
```

8.5. Upgrading Archives from Earlier ListProc Versions

The format of the DIR and INDEX files varies slightly with the ListProc version. The differences between ListProc 8.0 archives and earlier versions are described below.

Versions 6.0 and above

The file formats have remained the same since version 6.0, with the exception of the addition of the % character used for AFD/FUI information. As this is added automatically by ListProc, this should make no difference for upgrading.

Version 5.5

The -B, -p, -r and -D options are new. However, the functionality of all previous flags is the same, so it is not necessary to modify any of the files.

Versions 5.41 and lower

The -a option has been extended to accept paths to archives.

The DIR files for each archive now include the size of each of the parts of the archived files. The sizes should be included between the number of parts and the path name. For example, suppose the archive file "foo" has two parts, of sizes 300,000 and 12482 bytes, respectively. The corresponding line in the DIR file should look something like this:

```
foo 2 300000 12482 /fullpath/foo A file named foo
```

8.6. Subscribing to Files - AFD and FUI

ListProc allows users to subscribe to archive files in two ways. With Automatic File Delivery (AFD), users will be sent a new copy of the archive file each time it is updated. Users who simply wish to be informed that the file has changed can subscribe to File Update Information (FUI) instead. The syntax of the **AFD** and **FUI** commands varies slightly for regular users and the site manager.

Manager Syntax

```
[QUIET] AFD ADD|DELETE|QUERY  &
      {archive [/archive-password] [file-list]}  &
      [ {archive [/archive-password] [file-list]}  ] ... &
      user-list
[QUIET] FUI ADD|DELETE|QUERY  &
      {archive [/archive-password] [file-list]}  &
      [ {archive [/archive-password] [file-list]}  ] ... &
      user-list
```

The site manager syntax is identical to that of regular users, with the exception that the site manager can execute **AFD** and **FUI** request on behalf of users. The email addresses of all effected users are specified by a space-separated list. Furthermore, the manger need not specify archive passwords when issuing these commands through ILP.

Example
The following command will set users foo@bar.com and user@foo.com to receive notifications whenever the files in the **listproc** archive are updated:

```
QUIET FUI ADD { listproc } foo@bar.com  user@foo.com
```

Since **QUIET** is specified, neither use will be notified that this change has taken place.

```
fui review { listproc file1 } { cooking }
```

Get a list of users subscribed to the specified archives/files.

User Syntax

AFD ADD|DELETE|QUERY {*archive* [/archive-*password*] [*file-list*]}
&
[{*archive* [/archive-*password*] [*file-list*]}] ...
FUI ADD|DELETE|QUERY {*archive* [/archive-*password*] [*file-list*]}
&
[{*archive* [/archive-*password*] [*file-list*]}] ...

ADD|DELETE|QUERY
These specify the action that is to be taken:

ADD	Subscribes the user to the specified files or archives
DELETE	Unsubscribes the user
QUERY	Check to see if the user is subscribed

archive
This specifies the name of the archive to which the user wishes to act upon.

/archive-password
This specifies the archive password. This is only required for private archives.

file-list
This is a space separated list of files in the archive to which the user wishes to act upon. If no files are specified, the command will act upon all files in the specified archive.

Examples
The following command will subscribe the user to updates of **`ref.ps`** and **`user.ps`** from the **`doc`** archive, and to all files in the **`cooking`** archive:

```
AFD ADD { doc /pass1 ref.ps user.ps } { cooking }
```

This example assumes that **`doc`** is a private archive protected by password **`pass1`**. If you decide you no longer wish to receive automatic updates of **`ref.ps`**, you would send this command:

```
AFD DELETE { doc /passl ref.ps }
```

To find out which files in the **`doc`** archive you are subscribed to (via AFD or FUI), you would send these commands:

```
AFD QUERY {doc /passl}
FUI QUERY {doc /passl}
```

9. ListProc's Interactive Capabilities

Although sending email commands works well in many situations, there are occasions in which it would be nice to have more immediate feedback on the result of your request. For example, suppose you were trying to find the archive file that contained the interesting discussion of microbes used in rodent digestive systems you remember seeing on the MICKEY-BIOLOGY list a few months back. Unless you remember some of the exact wording used in the discussion, it may take you several trials before you come up with a useful match. Alternatively, suppose you are the server manager, and one of your most active lists has stopped processing mail because it has passed its MESSAGE-LIMIT for the day. It would be nice to be able to preempt all other server requests, and FREE the list, so it would be able to process mail again.

In both cases, the delay involved in waiting for the server to return your email makes dealing with the server more cumbersome. Fortunately, the Interactive ListProc (ILP) protocol allows you to communicate directly with the server, and hence avoid the inherent delays of email. Furthermore, since ILP requests are handled asynchronously with incoming email requests, you won't have to wait for ListProc to finish processing its incoming mail queue before getting to your urgent commands.

9.1. ILP Basics

How it Works

ILP uses a standard IP port to create a connection between the ListProc server process (**serverd**) and your ILP client. The fundamental communications between the client and the server consist of text based client queries and server responses. In fact, **serverd** handles the incoming commands by writing formatting them so they look like an email message, and piping them to the **listproc** program for processing. These messages appear in LPDIR/mbox along with all of the email messages the server has processed.

Logging In

There are several things involved in creating an ILP connection. When you first connect, you will be asked to identify yourself through your email address. This address will be written to messages as they are fed to **listproc**. Next, you are asked to provide a password. The password you provide determines the options used when **serverd** calls **listproc**. For example, if you use a subscriber password, a series of "-d *command*" flags will be used to disable commands (such as CONFIG or INIT) that are unavailable to subscribers. Hence, even if you are both the site manager and a list subscriber, you will need to log in with the site password in order to use manager commands - even with the correct password, these commands will be ignored because **serverd** will have disabled their use when you first logged in.

Sending Commands

The command syntax used with ILP is exactly the same as that used for email commands. The only difference between the two is that for you may abbreviate commands that require a password (e.g. CONFIG) by using "-" in place of the actual password. After it receives your command, **serverd** looks for a "-" in the place where a command word would normally be (essentially, it just looks for "-" surrounded by white space). The first such occurrence is replaced with your login password before the command is sent on to **listproc**.

Example
If you login with the password "churnel", the command

```
config cren-listproc - owners willoughby@hedgehog.net
```

would be translated as follows:

```
config cren-listproc churnel owners willoughby@hedgehog.net
```

Caution
ListProc does not check the semantics of the user command to make sure the first "-" appears in the position of a password. Hence, if you use a "-" in the wrong part of the command, ListProc will not hide your password when it responds to your message. For example,

```
config - foolist owners willoughby@hedgehog.net
```

will produce this error message:

```
Your request,
config churnel XXXXXXX owners willoughby@hedgehog.net
produced the following output:
No such list, XXXXXXX.
```

This could be problematic if you are working in an insecure environment, as your password is transmitted back to you in the clear.

User Privilege Levels

ListProc recognizes four user privilege levels for ILP, casual, subscriber, list owner, and site manager. The following table lists the commands that are allowed for each privilege level:

User Level	Available Commands
casual	lists, review, index, get, and search
subscriber	casual plus sub, unsub, set, and purge
list owner	All commands except init, purge
site manager	All commands

ILP User Privilege Levels

9.2. ILP Client Programs

Since ILP is based on text transfers, it is possible to connect to any ILP enabled ListProc server simply by telnetting to its ILP port. Since telnet clients are readily available for all platforms, this allows you to use ILP from any machine that can connect to your server.

CREN also offers a number of client programs that provide additional functionality by communicating directly with the server. The platforms currently supported are listed below. All versions are available via anonymous ftp from ftp.cren.net, in the /software directory.

UNIX
The UNIX ILP client, **ilp**, provides a text based interface to ListProc's interactive features, much like that available through telnet. This client has a couple of important advantages over telnet, however. First, since **ilp** need only communicate with ListProc, it is much smaller and uses less memory that telnet. More significantly, **ilp** allows the redirection of input and output (like that used by the UNIX shell) with >, <, and |. This provides a convenient mechanism for saving the results of commands, as well as for sending input from a file. A sample **ilp** session is included in Section 13.6 on page 88.

For your convenience, a pre-compiled version of **ilp** is included with the ListProc distribution packages. Additionally, the source code is included in the LPDIR/archives/ilp directory.

X-Windows and MS Windows
The X-Windows and MS Windows client (**xilp** and **winilp**, respectively) provide a graphical interface to ListProc's commands.

Macintosh
A Macintosh ILP client is currently under development. In addition to the ease of a graphical interface, this client will retain a list of subscribed lists and passwords between sessions, making access much more convenient. An interim release that includes only user commands should be available during the summer of 1996. The complete version, which will include the full range of owner and manager commands, should be ready shortly thereafter.

9.3. Controlling ILP Access

ILP access privileges can be controlled through the **`LPDIR/priv.host`** and **`LPDIR/unwanted.hosts`** files. These files are based on the **`/etc/hosts.allow`** and **`/etc/hosts.deny`** files common with UNIX tcp wrappers. When **serverd** senses an incoming ILP connection, it makes sure that connections are allowed from the remote host before starting the ILP session. If **`priv.host`** exists, it must contain either the hostname or the IP address of the remote host in order for the connection to be allowed. If there is no **`priv.host`** file, **serverd** will check to make sure that the incoming host is *not* listed in the **`unwanted.hosts`** file. If neither file exists, connections will be accepted from all hosts.

Each host in **priv.host** and **unwanted.hosts** is defined on a line by a line listing either the host's IP address or name. Wildcards can be used to specify entire domains.

Examples:
The following **priv.host** file would allow access only from the "foo.edu" domain and from the machine "freddy.munster.org":

```
.foo.edu
freddy.munster.org
```

If someone from "filbert.frubjous.net" is causing problems on your site, you could exclude them by adding this line to your **unwanted.hosts** file:

```
filbert.frubjous.net
```

If you want to allow access only from hosts listed in your **/etc/hosts.allow** file, you could simply create a symbolic link from **/etc/hosts.allow** to **priv.hosts**:

```
ln -s /etc/hosts.allow $LPDIR/priv.host
```

9.4. Running ILP Through inetd

Depending on the configuration of your ListProc server machine, you may wish to control ILP accesses by starting it with **inetd**. For example, if your ListProc server machine is located behind a firewall, but you wanted to allow ILP access from the outside, you would need to forward ILP connections through the firewall. Alternatively, you might want to use your tcp wrapper system to log ILP accesses.

ListProc uses coordinated file locking techniques to ensure the integrity of its data when they are accessed by multiple processes. Hence, you should *not* start **serverd** directly from **inetd** as this will not pass the appropriate semaphore information to any of the new processes.

The **gateway**[1] utility solves this problem by forwarding ILP requests to a specified host and port. The usage is as follows:

```
gateway remote-host remote-host-port
```

Example
Suppose you wish to forward ILP requests to port 1800 on your ListProc server machine, "host.your.domain." The corresponding entry in your **inetd.conf** would look something like this:

```
ulistproc stream tcp nowait root /home/server/gateway
/home/server/gateway myhost 1800
```

(The actual entry should be one line. It was wrapped to fit the formatting of this page.)

[1] The **gateway** utility can actually be used to forward almost any kind of TCP/IP traffic. See the full description of **gateway** on page 90 for more information.

10. Integrating Multiple Lists or ListProc Servers

ListProc provides a number of mechanisms for linking lists and ListProc servers together. Lists on different servers can be sewn together to form one logical list through ListProc's peering capabilities. ListProc can also be configured to forward commands intended for a list not handled by the local server to the server that actually handles the list.

10.1. Peer Lists

Peer lists are mailing lists that are subscribed to one or more other mailing lists. Each list in a peer group handles local distribution just as it would normally. Additionally, however, messages that originate in one of the peers are propagated to the other peers for distribution to their respective local subscribers. For every list message, only one connection is required to each of the other peers. Hence, peer lists can be used to distribute the job of sending out mail to a list. In particular, organizing each of the peer lists such that its subscribers are local to the server handling that peer can significantly reduce the time spent on (possibly slow) network connections to remote sites.

Peer lists are set up using the **`peer`** command included with the ListProc distribution. This creates an **`LPDIR/lists/LISTNAME/.peers`** file with information about the peer-ed list.

Note:
ListProc takes special measures with peer-ed lists both to prevent e-mail loops, and to ensure that the same message is not sent out to any of the peer lists twice. In order for these measures to properly *both the local and remote ListProc administrators must issue the appropriate peer commands*.

The syntax for the **`peer`** command is as follows:

```
peer LISTNAME peer-name peer-email peer-server-email
```

LISTNAME
The name of the local list. This should be in all capitals.

peer-name
This is the name of remote list which will be the peer of ***LISTNAME***.

peer-email
The full email address of the peer list.

peer-server-email
The full e-mail address of the ListProc server that handles the remote peer.

Example:
Suppose the ListProc managers at the Seattle and Taipei branches of Bogus University (BU) wish to set up their administration lists (adm-us@bu.edu and adm-tw@bu.edu.tw,

respectively) as peers, to reduce the delays associated with their slow trans-Pacific network connection.

The ListProc manager at bu.edu in Seattle would issue the following peer command:

```
peer ADM-US adm-tw adm-tw@bu.edu.tw listproc@bu.edu.tw
```

Meanwhile, in Taipei, the ListProc manager would use this command:

```
peer ADM-TW adm-us adm-us@bu.edu listproc@bu.edu
```

Once these two commands are issued, the connection is automatically set up.

Cautions:

- At most one list from a group of peers should be gateway-ed to any given USENET news group.
- Lists handled by the same server cannot be peers.
- A peered list should never be listed in the **`.subscribers`** file of one of its peers, as this will circumvent ListProc's built in loop detection scheme for peer lists. Peer lists should only be listed in the **`.peers`** file mentioned above.

10.2. Remote Lists

In addition to peer lists, ListProc provides two mechanisms for forwarding server commands intended for a non-local list to the server that actually handles that list. Information about individual lists can be explicitly stored in the **`LPDIR/remote.lists`** file. When ListProc receives a command intended for one of the lists in the **`remote.lists`** file, it automatically forwards the command on to the correct server. This feature is particularly useful for environments in which one organization uses multiple ListProc servers, as it frees subscribers from the worry of remembering which list is handled by which server.

The syntax of the **`remote.lists`** file is as follows:

```
remote list list-email server-email [hostname [port]]
#description
```

list
The name of the remote list.

list
The email address of the remote list.

server-email
The email address of the remote ListProc server.

hostname
The host name of the remote ListProc server. If the hostname for the remote ListProc server is included, ListProc will attempt to forward the command via the Interactive

ListProc Protocol (ILP), rather than sending email. If the ILP connection fails (if the remote host had disabled ILP, for example), ListProc will revert to sending email to the remote host.

port
The port to use for interactive ListProc sessions with the remote server. If no value is specified, the default of 372 is used.

description
A brief description of the remote list. Note that the pound sign (#) is required to separate the description from the previous parameters.

Example
The following line tells a local ListProc server to send commands for the list "fub" to the ListProc server at "fub.org":

```
remote fub fub@fub.org listproc@fub.org #Fub For Fun list
```

Since no hostname is included, all interactions with this server will be done via email, rather than ILP.

10.3. The Global ListProc Server

If a user sends a request for a list that is neither local to the server nor in the **`remote.lists`** file, it forwards this request to the Global ListProc server, as defined by the **global_update_sever** directive in the ListProc config file. The default config file sets this to CREN's Global ListProc server, listproc@listproc.listproc.net. The Global Server maintains an up-to-date list of published ListProc and Listserv lists. Thus, as long as the list is published, the Global Server will be able to the user's command on to the correct server for processing. (See page 44 for information on how to make a list published.)

11. Using ListProc With Other Information Services

One of the many splendors of the information age is the ability to cater to the individual tastes of users by presenting the same data in different ways. ListProc helps to achieve this goal through features that simplify integration with other information systems. Although they are not discussed in this manual, there are also ways of integrating ListProc's list archives and even user interface with World Wide Web services. We encourage site managers to discuss Integration issues of all kinds and share ideas on the CREN-LISTPROC mailing list.

11.1. News–List Gateways

ListProc allows full integration of email lists and USENET newsgroups through both one-way and two-way news-mail gateways. A one-way News-List gateway allows a ListProc list to receive postings from a particular newsgroup. A two-way gateway allows ListProc to both post news messages *and* receive postings from the news group. This can be particularly useful, as it allows users to choose whether they would like to interact through email, or through a news reader program.

Prerequisites

There are two prerequisites to setting up a News-Mail gateway :

- You must have access to a news server (the machine that will act as the news gateway) that is running *CNews* or *INNd*. For two-way gateways, the machine where the newsgroup resides must have adequate permissions to post articles to the server.
- You must either run **inews** or have access to mail-news gateway running appropriate software like **mail2news** or **gatenews**. (Your ListProc config file must reflect this choice through including either an "option post_mail" line if you are running **inews** or an "option gate_mail" line, if you are using a news-mail gateway.)

The "news" Command

The ListProc manager can easily set up a News-List gateway by invoking the **news** command (included with the system) from the ListProc server's home directory. This command will subscribes the newsgroup to the list. In order to prevent possible loops with the News-List gateway, the **news** command also stores information about the gateway in the **`LPDIR/lists/LISTNAME/.news`** file.

The syntax for the **news** command is as follows:

```
news LIST-ALIAS newsgroup news-email-address news-mode
```

LIST-ALIAS
The alias of the local list that will be linked with the news group. This should be in uppercase.

newsgroup
The name of the news group. (e.g. comp.os.linux)

news-email-address
This is the full e-mail address of the news source. The value of this field is only used if you are using a news-mail gateway (as defined by using the "option gate_mail" line in your ListProc config file). If you are using **inews**, simply use a dummy address in this field as a place-holder.

news-mode
There are two options for the news mode. "receive" specifies that the local list will receive news articles, but will not post list mail to the newsgroup. If the list should both receive news articles *and* post list articles to the news group, ***news-mode*** should be set to "send_receive."

Example:
Suppose you can send and receive news postings from the News-Mail gateway "news-gw@news.rubber.org". The following command establishes a two-way News-List gateway between the newsgroup "alt.gumby" and the "gumby-L@list.rubber.org" list:

```
news GUMBY-L alt.gumby news-gw@news.rubber.org send_receive
```

This command will create the file **`LPDIR/lists/GUMBY-L/.news`** with the following contents:

```
news-gw@news.rubber.org NOACK alt.gumby
```

This tells ListProc that mail coming from the address "news-gw@rubber.org" should be passed through to the list. This also tells ListProc to forward the list mail on to that address.

In order to make sure the news postings are sent to the list, you will also need to modify the configuration of your news software. For example, if you are using C-News, you will need to add the following line to the C-News **`sys`** file (usually found in **`/usr/lib/news/sys`**):

```
mail.to.gumby-l@list.rubber.org:
alt.gumby/all::news-gw gumby-l@list.gumby.org
```

(Note: This should be one line. For the sake of readability, it is shown here as two.)

Caution:
Before distributing an incoming message to a list, ListProc checks to see if the sender has permission to post to that list. Since news postings passed through a news-mail gateway retain the "From:" header from the news posting, ListProc may not recognize the sender, and hence reject the message. An easy way around this is to configure the list to allow postings by anyone (i.e. set it to SEND-BY-ALL). Another option is to configure the news-mail gateway to include a "Resent-From:" header with the email address of the gateway. Since ListProc checks the "Resent-From" header before the "From" header when determining the identity of the sender, this will allow ListProc to correctly identify the message as coming from the news group, and hence to accept it for distribution to the list.

11.2. Anonymous FTP

Running ListProc on the same machine as an anonymous FTP server allows you to very simply provide both anonymous FTP and ListProc archive access to the same files. This allows you to use ListProc to distribute files via email to people who can't use FTP directly. Furthermore, if you enable AFD and FUI (see page 67) for certain archives, users will have the option of receiving these files automatically each time the FTP archive is updated. Conversely, by telling ListProc to archive list mail in an FTP-accessible directory, the latest list archives will always be accessible to FTP users.

There are two prerequisites for integrating ListProc with FTP:

- You must have an ftp server installed on your ListProc server. (Consult your system's **ftpd** man page for information on how to set up an anonymous ftp server.)
- The ListProc server account must have write permission in the directories in which you wish to allow both ftp access and ListProc access.

File Permissions

In order to make sure that FTP users can actually access files from shared ListProc-FTP archives, it is essential that you tell ListProc what file permissions it should use for its archives. This is configured by setting the ULISTPROC_ARCHIVES_UMASK environment variable before starting the server (see page 13). Setting this variable to 022 will prevent ListProc from making any of the files group or world writeable, but will allow them to be world readable (and for directories, world executable as well).

Archiving List Mail In An FTP directory

This can be accomplished with one use of the CONFIG command. Suppose you would like the archives from the "beach-nut" list to be accumulated in the **`/ftp/pub/lists/beach-nut`** directory. You could accomplish this by sending the following command to ListProc:

```
config beach-nut manager-pw archive /ftp/pub/lists/beach-
nut
```

(Note that the other arguments to the archive command can be left out here, since the only thing we are interested in changing is the directory.)

Note
Setting the position of the list archive will only effect where ListProc *adds* files to the archive. In order to make the other files accessible via ftp, they will need to be moved by hand.

Adding Files From An FTP Directory To A ListProc Archive

Files currently available via FTP can be added to a ListProc archive with the **farch** command line utility (see page 64).

Example
Suppose we wanted to create a ListProc archive that included the files the public ftp directory **`/ftp/pub/test`**. The farch command line and output might look something like this:

```
unix% farch -Z -a test -d /ftp/pub/test /ftp/pub/test/*
New archive test:
/ftp/pub/test/file1:
        - file not split, not compressed
        - archived in test
        - directory: /ftp/pub/test
/ftp/pub/test/file2:
        - file not split, not compressed
        - archived in test
        - directory: /ftp/pub/test
/ftp/pub/test/file3:
        - file not split, not compressed
        - archived in test
        - directory: /ftp/pub/test
unix%
```

Note the use of the "-Z" flag, to tell farch not to compress the archive files. If this is *not* used, **farch** will automatically compress all of the files in the /ftp/pub/test when it records them in the ListProc archive.

Note
ListProc will not automatically update the file information for each file in the archive. Thus, if you use FTP to update files in a shared ListProc-FTP archive, the changes will not show up in the ListProc archive index. Also, new files added through FTP will be invisible in the ListProc archive. To update the ListProc archive, simply use the **farch** command just as you would if you were creating the archive for the first time.

11.3. Gopher and WAIS

Like anonymous ftp, gopher can be set up to provide access to your list archives. Also like anonymous ftp, you must place the archives you want to make available via gopher in the directory tree that your gopher files are running. ListProc's account (server) must have read-write permissions to this directory.

When using gopher, it is important that you do not compress archives. By maintaining uncompressed archives, your gopher server will automatically pick up that the archives are Unix mailboxes and present them to the user as a list of individual mail items. (Gopher to cren.org to see an example of this.) Every new posting on a list where there is archiving is automatically added to the gopher menu. If you compress archives, this type of presentation is not possible.

It is also possible to WAIS index the mail logs. To do this, you must compile and install your gopher server with the WAIS capability as described in the gopher installation instructions. For an example of how this is done, again gopher to cren.org.

Unlike gopher, WAIS is not automatically done and must be updated via **cron**. WAIS has flags to automatically pick up mailboxes and index them appropriately. The following is an example:

```
cd /archive_dir/.archivename
find /archive_dir/archive-name -type f -name “log*” -print
| grep - v ‘/\.’ | waisindex -a -d archiveindex -t
mail_or_rmail -stdin
```

In this example, first go to a directory called **.archivename**—the period in front of the name makes this directory invisible to gopher. This directory is where all the waisindex information is stored. It must be read-writeable by the ListProc account. It can be set up with permissions 600. Then use the find command to pass WAIS a list of your archives. The -t flag to WAIS tells the server that this list being passed to it is a mail_digest (i.e. UNIX mbox). The -a flag is used to reindex only those files which have changed. A full indexing should be done approximately once per week.

For every archive that you want to make WAIS searchable, you should run this nightly out of the ListProc’s user-id cron entry to reindex the archives.

There is another caveat with WAIS and gopher. For security reasons, the gopher server runs chroot-ed. The problem with WAIS Indices is that they store the full path name inside the inverted index. Thus, if your documents are in:

```
/usr/local/archives/listname
```

that entire path will be kept in the index.

However, when using *chroot()*, a whole portion of the path is chopped off and the directory /usr/local/archives effectively becomes root, ‘/’.

To circumvent this problem, you can use a symbolic link to make the removed directory structure inside the archives directory. In the case above, you would do the following:

```
cd /usr/local/archives
mkdir usr
cd usr
mkdir local
cd local
ln -s / archives
```

Then, when the path name **`/usr/local/archives/listname`** is looked up, the symbolic link will put you in the correct place. Also, so that your gopher users do not see this new directory, add the following line to the **`gopherd.conf`** file:

```
ignore: usr
```

12. Customization and Maintenance Tips

A bit of care in fine tuning your ListProc setup can save considerable time and effort by reducing the number of daily maintenance tasks and helping to organize debugging information for easy access in case problems arise.

12.1. Server Message Files

Many of the basic messages ListProc uses when giving information to subscribers can be tailored to suit the particular needs of your site. Although the generic versions included with the ListProc distribution should be fine in most cases, slight modifications of these may help your users better understand how to use ListProc, and thus reduce the number of questions you must answer by hand.

ListProc's site configurable messages are included as simple shell scripts in the **`LPDIR`** directory. When a particular message is required, ListProc simply invokes the correct script, and uses the script's output as the body of the outgoing message. Thus, updating a message is simply a matter of editing the text of the associated script file. The following message scripts can be modified:

`LPDIR/welcome.global`
`LPDIR/welcome.live`
`LPDIR/signoff.global`
All of the help files in **`LPDIR/help`**

12.2. Creating additional help topics

ListProc also allows you to create site specific help topics, in addition to those included with the system. The path names used to generate responses to a particular help query are defined in the **`LPDIR/help/TOPICS`** file. Thus, to add a new help topic to the system, just edit the **`TOPICS`** file with your favorite text editor, and specify the name of the help topic, and the path to the script that generates the output. Each line in the **`TOPICS`** file should be formatted as follows:

```
help-topic   full-path-to-script
```

12.3. Mailers

Your choice of a Mail Transfer Agent is a critical factor in determining the overall performance of your ListProc server.

At the very least you should install the latest version of UCB **sendmail** (8.6.13 as of this writing) and the Berkeley *db library*. The new **sendmail** with its expanded database capabilities and limited multi-queuing will give you some strong performance

enhancements. Both packages are available via anonymous ftp from ftp.cs.berkeley.edu in the **`/ucb/sendmail`** directory.

For high volume sites, the University of Toronto's **zmailer** might be a better choice. This mailer has its rewrite rules optimized for SMTP only hosts. **zmailer** is a large and complex system but it seems to pay off for large sites. You can obtain **zmailer** via anonymous ftp from ftp.cs.toronto.edu in **`/pub/zmailer`**.

Another alternative is **MMDF**; its multi-queued nature will give you enhanced performance over sendmail.

13. ListProc Commands and Utility Programs

The ListProc distribution includes a number of utility programs. Several of these have been discussed in previous sections. For completeness, however, all will be mentioned here.

13.1. Main Server Programs

The following files are all used internally by ListProc. In most cases, the only commands you will need to call explicitly are **start**, **stop**, and **queued** The others are described here to help you understand ListProc's logs and warning files, and to allow you to use the UNIX **ps** command more effectively to determine what ListProc is doing at any given time.

!!Caution!!
It is **imperative** that these commands be kept in the root directory of your ListProc installation. Many of the commands expect to be able to call the other commands as "LPDIR/command-name". Hence, simply including them in a directory in your path will not work. Furthermore, since **catmail** is called by your local MTA, it cannot rely on the LPDIR environment variable to determine ListProc's location. Hence, it must assume that it the directory from which it was called is the ListProc directory.

start

The **start** command is used to start the ListProc system. The first thing **start** does is to check if there are any old ListProc processes lying around. If there are, it sends them the INT signal, and waits for them to exit. Once all leftover processes have been removed, **start** spawns **serverd** and then exits.

start accepts the following command line options:

-k
Kill all existing ListProc processes, without spawning a new **serverd** process. This can be used to cleanly bring down the server.

-c
By default, **start** will prompt you to confirm that you actually want to kill each remaining ListProc process. This option turns off this prompting, so that **start** can be called from a script. This is particularly useful if you wish load ListProc at boot time from your system startup files.

stop

This is actually just a symbolic link to **start**. When the **start** program notices that it was called as "stop", it turns on the -k flag. Hence, "**stop**" is equivalent to "**start -k**".

catmail

The **catmail** utility is used along with MTA aliases to send incoming email to the correct message folder. The command line arguments specify the folder to which **catmail** should append the message. The message itself is read from **catmail**'s standard input.

The command line options are as follows:

-L *list-name* | -l *list-name*
Specify that the message is intended for the list ***list-name***. If neither the -e or -o flags are used, the message will be appended to LPDIR/lists/list-name/mail.

-e
This flag is used in conjunction with the -L flag, and specifies that the message should be added to the list's error file, rather than to the list's standard incoming mail file.

-o
Like -e, this flag is used in conjunction with the -L flag. This instructs **catmail** to send the message directly to the owner(s) rather than appending it to a particular file.

-r
This specifies that the message is intended for the ListProc server account, and that it should be appended to LPDIR/requests.

-f
This instructs **catmail** to reformat the message before it sends it to its final resting place. Specifically, **catmail** will indent any lines that begin with "From " with a right angle bracket (>) to make sure the message is in UNIX mbox format.

In order for **catmail** to run correctly, it must be owned by the server account, and setuid. This will ensure that the access permissions on ListProc's message files remain correct, so that ListProc can access them correctly.

If the incoming message is for a moderated list and not from the moderator, append an INTERNAL_NOTIFY request to the requests file so that ListProc will later notify the list owner, soliciting his approval for posting the message.

Examples:
The following examples all show how **catmail** can be used from your MTA's alias file. In all cases, "LPDIR" should be replaced with the full path to the server directory.

To append to the system's requests file:

```
ListProc: "|LPDIR/catmail -r -f"
```

To append to a list's mail file:

```
list_alias: "|LPDIR/catmail -L LIST_ALIAS -f"
```

To append to a list's errors file:

```
owner-list_alias: "|LPDIR/catmail -L LIST_ALIAS -f -e"
```

To send to owners:

```
list_alias-request: "|LPDIR/catmail -L LIST_ALIAS -f -o"
```

serverd

The **serverd** program controls the flow of ListProc's execution. One **serverd** process spawns the **listproc** and **list** processes that process and respond to ListProc's incoming mail. If ILP has been enabled, another process listens for incoming ILP requests. One additional **serverd** process will be spawned for each interactive session currently active. (For more information on ILP, see Chapter 9, ListProc's Interactive Capabilities, on page 69.)

This command should not be run by hand. Instead, use **start**, which will read the global config file, and call **serverd** with the options specified in by the **serverd** config file directive (see page 23).

list

The responsibility for processing list specific mail falls to the **list** command. Incoming list mail (messages in LPDIR/lists/listname/mail) are processed and sent out to the list's subscribers. Error messages sent to LPDIR/lists/listname/errors are also processed, and the appropriate action taken (see the description of the **error_analysis** config file directive on page 33 for more information). **list** also accumulates digests and list archives if these have been enabled in the list's config file.

There is no need for you to call the **list** program explicitly, as it will be automatically called by **serverd**. However, when using **ps** to examining the processes running on your system, it may be useful to understand the command line options that **list** uses:

-L
This specifies the name of the list to be processed.

-S *semid*
This gives the ID of the semaphore that **list** should use to communicate with its parent and sibling processes. This flag is new to ListProc 8.0.

-E
The -E flag informs **list** that it should process the list's error mail. If this flag is not specified, only the list's regular mail will be processed.

listproc

The **listproc** command parses and responds to all incoming server commands (the messages in LPDIR/requests). Hence it is responsible for adding and removing subscribers, modifying subscriber settings, sending list statistics, sending archived files, etc.

As with the **list** command, there is no need to call **listproc** by hand. For your reference, **listproc**'s command line options are as follows:

-S *semid*
Specify the ID of the semaphore that **listproc** should use to communicate with its parent and sibling processes.

pqueue

When using the **system** mail method (see page 28), the **list** program writes the outgoing messages to temporary files, and calls **pqueue** to process and send them to your MTA. Occasionally, your MTA host will reject a message for some reason. (If the sender's address is malformed, for example.) When this occurs, the site manager is notified, and the offending message is saved in the LPDIR/mqueue directory. After examining (and hopefully fixing) this message, you can call **pqueue** to process the mail queue and re-send the message.

For more specific information on ohw to use **pqueue**, see the **pqueue** man page in the **`LPDIR/doc`** directory.

farch

The **farch** command is discussed extensively in Chapter 8, "Archiving Files". (See "Command Line Archive Management - the *farch* Utility" on page 64.) Briefly, **farch** can be used from the command line to maintain ListProc archives. Coincidentally, **farch** is also used internally by the email ARCHIVE command (see page 61).

13.2. Configuration Utilities

LPDIR/setup

This is the system setup script. Its operation is described in some detail in section 4.1 on page 11.

LPDIR/news

This script is used to add a newsgroup to a list. For a complete description of how to use the **news** command, see Section 11.1, "News–List Gateways" (page 76).

LPDIR/peer

Script to add a peer server. The **peer** command is described in detail in Section 10.1, "Peer Lists" (page 73).

13.3. Maintenance Utilities

dbglpfiles

The **dbglpfiles** command provides a simple mechanism for checking whether or not ListProc's **.aliases**, **.ignored**, and **.subscribers** files are properly formatted. For **.alias** and **.ignored**, **dbglpfiles** enters into a command line environment that allows the user to test out addresses and see how ListProc would deal with them. For .subscribers files, it performs a simple format check, and prints out any errors that it finds.

For a full description of **dbglpfiles**, see the included man page, **LPDIR/doc/dbglpfiles.1**.

cycle_logs

Without some sort of intervention, ListProc's log, **.warning**, and **mbox** files will continually grow, eventually filling up your hard drive. While these files are useful for tracking down problems, their contents clearly needn't be saved forever. At the same time, they should be kept around for at least a few days, to be sure that enough debugging information is around should a problem arise.

This problem can be easily solved by renaming the log, .warning, and mbox files on a regular schedule and deleting the saved files after they have been saved for a specified length of time. The **cycle_logs** script automates this process. Current files are given a postscript of ".0", and the postscripts of previously saved files are incremented. Files with a postscript of ".N" are deleted, where "N" is the maximum number of files to save.

We recommend that you set up **cycle_logs** as a **cron** job to automatically run once a day. The script comes configured to save seven days worth of files. If you wish to change this, you should edit the script and change the "DAYS-TO-SAVE" variable. Also, by default, **cycle_logs** will look use the LPDIR environment variable to determine where to look for the log files. Depending on your system configuration, however, the LPDIR environment variable may not be set when **cron** runs **cycle_logs**. If this is the case, you will need to edit the script, and manually enter the path to your ListProc directory in the scripts "CYCLE-DIR". Finally, if you are using **syslog**, you will need to set the script's "LOG" variable to the full path and name for your list's log files.

Example:
If your ListProc server was installed in **/home/server**, the following entry in server's **crontab** file would run **cycle_logs** every day at 1:00am:

```
0 1 * * * * /home/server/utils/cycle_logs
```

Consult your system's man pages on **cron** and **crontab** for more information on scheduling commands with **cron**.

13.4. Message Digests with md5

In checking for duplicate messages, ListProc makes extensive use of message digests. By default, ListProc uses the UNIX **sum** command, which produces 16 bit sums. Although ListProc retains only the most recent 2000 message digests for any list, when using **sum**, it is all too easy to wind up with different messages that have the same checksum. In this case, ListProc would reject the message as a duplicate, even though the messages were actually different.

To solve this problem, ListProc offers the option of using RSA data Security Inc.'s **md5** message digest program instead. (Technically, "md5" refers to the message digest algorithm. As with so many other things, however, the term has been overloaded, and is also used to refer to a common implementation of the algorithm.) The md5 algorithm creates much longer 64 bit digests. Additionally, the algorithm has been optimized to take advantage of some of the uniformities of ASCII text, so the likelihood of different messages having the same sum is very small.

Obtaining and Compiling md5

You can obtain a source code version of md5 from the /software/MD5 directory of CREN's anonymous ftp server, ftp.cren.net. The source should compile without difficulty on most platforms, but you should refer to the included README to see if there are any specific caveats for your system. Typing "make" will build the **md5** command, and "make test" will check to ensure the build was successful.

Using md5 with ListProc

There are two requirements for using md5 with ListProc. First, the **md5** command must be somewhere in the ListProc server's path. Choose the directory into which you wish to install **md5**, and check to see if it is in the server's path. (To check the server's path, login to the server account, and type "echo $PATH".)

13.5. System Message Utilities

ListProc includes the ability to customize many of the responses it gives to user queries. These are described at some length in "Server Message Files" on page 81.

13.6. The UNIX ilp Utility

Ilp requires that the site manager, list owner and regular users have IP connectivity. It is a command line interface that you enable simply by typing:

```
ilp hostname port
```

To connect to CREN's ListProcessor, the command line would be as follow:

```
ilp yukon.cren.org
```

You do not have to specify a port address since we are using the standard port for ilp. Ilp will present the following prompt:

```
Hit ^\ (control-backslash) to abort at any time.

CREN interactive ListProcessor, (c) 1993-94 by CREN.  All
rights reserved.
Version 8.0 Mon Jul 18 14:23:30 1994
Maximum connection duration is 1800 seconds.

Welcome!

The set of requests you may issue during this session
depends on the privileges
assigned by the system.  You may log in as a:

-Subscriber: provide your e-mail address and password as
subscribed to one of the local lists.

-Owner: simply provide your e-mail address and your list's
password.

-Manager: provide your e-mail address and the system's
password.

-Other: simply hit return at the prompt below.

e-mail address [none]:
```

At the e-mail address prompt, you type in the e-mail address by which you are known to the system. This could be a list owner's address, a site manager's or subscriber's address. If you enter no e-mail address, you will have the minimum privileges on the system. The next prompt:

```
Password:
```

is your access password. Again this could be your subscriber's, list owner's or site manager's password. If you do not enter a password, you will have minimal privileges on the system. This password will not be echoed to the screen, and **you do not need to type the password again after initial authentication**. For all future commands, you can substitute a hyphen (-) for the password, i.e.

```
conf list-name -
```

and

```
conf list-name password
```

are equivalent.

As a list owner, it provides a handy way of getting information about your list and pipes and it allows redirect. The angle brackets (< >) indicate the directional flow of the information. For example to find out if jdoe@foo.bar is subscribed to the list, the following will work:

```
review list-name | grep -i jdoe
```

```
conf list-name - > list-name.config
```

The first command will return only those subscribers where the string *jdoe* (UPPER and lower case) appears. The second command will return the list configuration (if you have logged in as a list owner) and redirect the output to a file, thus allowing you to edit this file and send it back to ListProc with some configuration changes as such:

```
conf list-name - < list-name.config
```

Sample ILP Session

```
unix% ilp list.cren.net
Trying 204.153.50.13 ...

Hit ^\ (control-backslash) to abort at any time.

CREN interactive ListProcessor(tm), (c) 1993-96 by CREN. All
rights reserved.
Version 8.0 Fri Jul 12 15:37:07 1996
Connection idle timeout is 600 seconds.


Welcome!

The set of requests you may issue during this session depends on the
privileges assigned by the system. You may log in as a:
- Subscriber: provide your email address and password as subscribed to
one of
  the local lists.
- Owner: simply provide your email address and your list's password.
- Manager: provide your email address and the system's password.
- Other: simply hit return at the prompt below.

email address [none]: listmgr@cren.org
Password:
You have logged in with manager privileges; you may not issue 'execute'
requests.

request> version
ListProcessor(tm), version 8.0 -- ListProcessor(tm) by CREN
Revision level: 8.0/960604/0
Last update: June 4, 1996
Manager: listmgr@cren.org
ListProcessor(tm) address: listproc@listproc.net
request> quiet add explore - foobar@cren.net Mr. Foo Bar
Your request: QUIET ADD EXPLORE XXXXXXX foobar@cren.net Mr. Foo Bar
produced the following output:
User foobar@cren.net was successfully subscribed to list
explore@list.cren.net.
request> quit
ListProcessor(tm) closing connection.
Connection closed.
unix%
```

14. ListProc Flow of Control

Simultaneous processing is performed on as many lists as there are threads defined in the config file. Hence, if your config file contained "threads system_wide 10", ListProc would be able to concurrently process mail from 10 different lists.

The spawning of threads is still subject to delays specified by the "frequency" directive in the config file. For example, suppose your config file contained the line "frequency 0 30". In ListProc 7.X, the flow of list processing would look like this:

process list 1 (regular mail, digests, error mail)
process server mail
process list 2 (regular mail, digests, error mail)
process server mail
...
process final list (regular mail, digests, error mail)
process server mail
wait 30 seconds
repeat

ListProc 8.0 does the following instead:

spawn thread for list 1 (regular mail, digests)
spawn thread for error mail (if error mail, and no error thread already running.)
spawn server mail thread (unless one already exists)
spawn list 2 thread (regular mail, digests)
spawn error thread (if error mail, and no error thread already running.)
spawn server mail thread (unless one already exists)
...
spawn final list thread (regular mail, digests)
spawn error thread (if error mail, and no error thread already running.)
spawn server mail thread (unless one already exists)
wait 30 seconds
repeat

Notice that in 8.0 ListProc does not need to wait for the processing of one list to finish before it goes spawns a thread to process the next list.

Within each list, processing is still serialized: regular mail and digests are never handled concurrently for a given list. As noted above, the error processing has been taken out of the list processing loop. Error mail is processed by the first available thread after error mail arrives.

15. Example Installation Session

The following is an example of upgrading from ListProc 7.x to ListProc 8.0.

```
unix% whoami
server
unix% pwd
/home/server
unix%
unix% install.listproc

                          ListProcessor(tm) version 8.0
                          -----------------------------

                            Copyright (c) 1993-96 by
           the Corporation for Research and Educational Networking (CREN)

Moving help/ to Ohelp/
Moving archives/ to Oarchives/
Moving config to Oconfig
Moving owners to Oowners
Moving .ignored to O.ignored
Moving .aliases to O.aliases
Moving unwanted.hosts to Ounwanted.hosts
Moving welcome.live to Owelcome.live
vMoving welcome.global to Owelcome.global
Moving signoff.global to Osignoff.global

Extracting new system ...

./
makefile
setup
ulock
config
owners
archives/
archives/ilp/
archives/ilp/INDEX
archives/ilp/DIR
archives/ilp/ilp.h
archives/ilp/ilpp.h
archives/ilp/ilp.1
archives/ilp/ilp.c
archives/ilp/makefile
archives/ilp/ilp.nr
archives/doc/
archives/doc/summary-of-requests
archives/doc/INDEX
archives/doc/DIR
archives/listproc/
archives/listproc/DIR
archives/listproc/INDEX
doc/
doc/ILPP
doc/catmail.1
doc/farch.1
doc/ilp.1
doc/queue.1
doc/serverd.1
doc/start.1
doc/ilp.nr
doc/farch.nr
```

```
doc/catmail.nr
doc/serverd.nr
doc/queue.nr
doc/start.nr
support/
support/ILPP
.aliases
.ignored
news
peer
queued
help/
help/TOPICS
help/add
help/afd_fui
help/alias_ignore
help/approve
help/archive
help/configuration
help/delete
help/discard
help/edit
help/fax
help/general
help/get
help/hold_free
help/index
help/information
help/initialize
help/listproc
help/lists
help/live
help/purge
help/put
help/recipients
help/release
help/reports
help/run
help/search
help/set
help/statistics
help/subscribe
help/system
help/unsubscribe
help/which
.awk
.stats
.grep
welcome.live
welcome.global
signoff.global
system-test
unwanted.hosts
gateway
listproc
list
serverd
start
tlock
farch
pqueue
catmail
ilp
rev
fwin
semset
dbglpfiles
```

```
Restoring old help... done; new help in Nhelp
Restoring old archives... done; new archives in Narchives
Restoring old config... done; new config in Nconfig
Restoring old owners... done; new owners in Nowners
Restoring old .ignored... done; new .ignored in N.ignored
Restoring old .aliases... done; new .aliases in N.aliases
Restoring old unwanted.hosts... done; new unwanted.hosts in Nunwanted.hosts
Restoring old welcome.live... done; new welcome.live in Nwelcome.live
Restoring old welcome.global... done; new welcome.global in Nwelcome.global
Restoring old signoff.global... done; new signoff.global in Nsignoff.global

Once you have studied the new system files, you should run "setup"
for the final setup.

You may now remove src.tar and ./install.listproc
N.aliases
N.ignored
Narchives
Nconfig
Nhelp
Nowners
Nsignoff.global
Nunwanted.hosts
Nwelcome.global
Nwelcome.live
archives
catmail
config
dbglpfiles
doc
example.install
farch
fwin
gateway
help
ilp
install.listproc
listproc
makefile
news
owners
peer
pqueue
queued
rev
semset
serverd
setup
signoff.global
src.tar
start
support
system-test
tlock
ulock
unwanted.hosts
welcome.global
welcome.live
unix%
unix% system-test
              System suitability test for ListProcessor(tm) 8.0

                        Copyright (c) 1993-96 by
        the Corporation for Research and Educational Networking (CREN)

ListProcessor(tm) is available on the following platforms:

SUN IBM SGI DEC HP Convex Stardent KPC NeXT SCO Apollo Sequent DG i860 OSF i386
```

```
Platform is SUN

--- Looking for required utilities:
echo: found
cut: found
paste: found
wc: found
cat: found
tr: found
md5: found; setenv MD5 1 in your .cshrc file, or MD5=1;export MD5 in your .profile
awk: found
mv: found
cp: found
rm: found
grep -i: works OK
uuencode: found
compress: found
zcat: found
uncompress: found
tar: found
egrep: found
tail: found
uptime: found
nslookup: found
ps: found: BSD version
chmod: found
chown: found
sort: found
uniq: found
inews: found
UCB mail: found; define 'ucb_mail /bin/mailx' in the config file.


--- Testing the system: It's got System V flavor
system() works OK
*** Contact CREN for more information on your SUN system

unix%
unix% setup

                ListProcessor(tm)/ListProc(tm) system setup script
                ---------------------------------------------------

                                Version 8.0

                            Copyright (c) 1993-96 by
          the Corporation for Research and Educational Networking (CREN)

The system will now be installed under /home/server;
if you want to install it under another directory, you may redefine the LPDIR
environment variable now, or specify another directory in the command line.
Proceed [y]? y
Remaking ./setup
Remaking ./news
Remaking ./peer
Remaking ./queued
Remaking ./Narchives/ilp/INDEX
Remaking ./Narchives/ilp/DIR
Remaking ./Narchives/ilp/ilp.1
Remaking ./Narchives/listproc/INDEX
Remaking ./help/TOPICS
Remaking ./help/listproc
Remaking ./Nhelp/TOPICS
Remaking ./Nhelp/listproc
Remaking archives/ilp/INDEX
Remaking archives/ilp/DIR
Remaking archives/doc/INDEX
```

```
Remaking archives/doc/DIR
Remaking archives/listproc/INDEX
Remaking archives/foolist/INDEX
Remaking archives/foolist/DIR
Remaking archives/foolist/CONFIG
Creating directory /home/server/lists
Creating directory /home/server/mqueue
Creating directory /home/server/tmp

Now serverd will have to be setuid to root if it is to run in interactive mode
and use a privileged port. Please enter the root password below if so:
Password: <your root password here>

NOTICE: the canned archives/ilp directory contains everything a user will need
for live connections; you may wish to copy it to your own archives now.

ListProcessor(tm) System ready.

unix%
unix%
```

16. Sample Config File

```
#!/bin/sh
#                 @(#)config  5.11 CREN 96/05/04
#
#                 Copyright (c) 1993-96 by
#     the Corporation for Research and Educational Networking (CREN)
#
# All rights reserved. This software comprises confidential
# information of CREN and may not be used, copied or made available to
anyone,
# except in accordance with the license under which it is furnished.
#
# SCCS file: /usr/SCCS/usr/server/s.config
#

# Example config file. The # character is used for comments. An & character
# followed immediately by a newline continues the command on the next line.

# Define your organization:
organization CREN

# Define the system's manager address:
manager foo@cren.net

# Define the system's master password (known to the manager only):
password secret-password

# Define your server's email address and optional command line arguments:
server listproc@cren.net host.cren.net    # -b get

# Define the global query server for all lists worldwide:
global-query-server listproc@listproc.listproc.net listproc.listproc.net 372

# Define the master server for collecting global list information, and the
# time to send a list of published lists daily:
global-update-server global-update-server@listproc.listproc.net 04:00

# Define the server daemon's (serverd) command line options:
serverd -i 1800 -l 3 # -p 1800 -c 10

# Tell the server when to batch requests:
batch 8 20   # batch requests between 8 am and 8 pm

# Tell the daemon how often to check for mail or requests (in seconds); if
the
# first arg is 0 (look for mail all the time), a sleep time can be specified
# so that serverd will rest at the of the scan of all lists (to reduce CPU
# usage):
frequency 5
# frequency 0 30
```

```
# Define the maximum number of processes to be used for mail distribution;
# default is 1; max is 255
threads system_wide 15

# Tell the daemon how often to process archives and perform AFD/FUI:
# deliver_files frequency [when]
deliver_files hourly 30       # Deliver hourly on the half mark
# deliver_files daily 8:30    # Deliver every day at 8:30 am
# deliver_files weekly Monday # Deliver every week on Mondays
# deliver_files monthly       # Deliver the first day of every month

# Define the listproc's name (Comments for the From: line):
comment server "CREN ListProcessor(tm)"

#Define the default archive directory for lists (default is
$LPDIR/archives/list-alias):
#default_arch_dir /ftp/mailing-lists

# Define the contents of the Precedence: header line.
# If not defined, no Precedence: line is added:
#precedence bulk  # Usually bulk, junk, first-class or none

# Tell the server about certain restrictions:
restriction 4      # certain requests may not be serviced abover this # of
users

# Define global limits:
limit message 65536     # reject messages longer than 65536 bytes
limit files 65536 # split files longer than 65536 bytes

# Tell the server how many recipients to pack together in a single message:
multiple_recipients 30

# If posting to newsgroups, define where inews resides (default
/usr/lib/news/inews):
# inews /usr/bin/inews

# Define system-wide options:
#option post_mail # Post to news; other choice is gate_mail
option gate_mail
option ignore_invalid_requests      # Ignore a request if it cannot be
recognized
#option relaxed_syntax        # Ignore extra arguments to requests
#option multiple_listprocs    # For sites running 2 or more copies
#option readonly_subscribers_files # Disable PUT ... SUBSCRIBERS by owners
# The following determines what kind of MIME digests are to be distributed;
# if the option below is defined, the system sends multipart/digest MIME
# digests where the preamble is a plain/text message and dashes are used for
# separators according to RFC 1153 (this RFC is experimental); otherwise, the
# default behavior is to send multipart/mixed with embedded multipart/digest
# MIME digests; we do not recommend the definition of this option.
#option rfc1153_compliant_digests
```

```
# Tell the system how to send mail. "system" is the recommended method:
mailmethod system
# mailmethod telnet
# mailmethod binmail
# mailmethod rmail /bin/rmail
# mailmethod sendmail /usr/lib/sendmail -ba
# mailmethod sendmail /usr/ucblib/sendmail -ba
# mailmethod sendmail /usr/lib/sendmail -t (IDA)
# mailmethod env_var LOGNAME /usr/ucblib/sendmail -ba
# mailmethod env_var LOGNAME /bin/mail
# mailmethod env_var LOGNAME /usr/lib/sendmail -t
# mailmethod env_var LOGNAME /bin/rmail
# mailmethod env_var LOGNAME /usr/bin/mail

# Define the host to connect for sending mail, and an optional port (def.
25):
mta_host localhost
# mta_host localhost 2000

# Set SMTP HELO argument:
helo_arg host

# Define blackout addresses (regular expressions):
mailer_daemon '[  ]*test$|^uucp|mrgate|mmdf|^smt.*|^mal@|^admin|^root|&
majordomo'

# Define blackout subjects (regular expressions -- notice the & continuation
# character which is supported throughout the config file):
#susp_subject 'remove[  ]+me|remove[      ]+my|&
delete[     ]+me|delete[      ]+my|add[   ]+me|add[   ]+my|&
^[    ]*test[     ]*$|^[      ]*test.+ignore|please[  ]+ignore|&
Confirmation[     ]+of[       ]+your[     ]+message|Receipt[
      ]+Confirmation'

# Turn on error analysis, set level and grace period (in seconds):
error_analysis 9 259200

# Turn on syslog reporting, set facility:
# syslog LOG_USER # LOG_DAEMON, LOG_MAIL, LOG_LOCAL0-7

# Ignore these extra requests (regular expressions):
ignore_requests 'OFFICE[      ]+MEMO|^[   ]*-|&
^[    ]*[A-Z][A-Z0-9-]+:[     ]*.*|^\>|^\]|^\||^#|^%|^\}|^\$|&
this[       ].+[  ]test|please[     ]ignore'

# Turn on BSD_MAIL and define the path to it:
ucb_mail /bin/mailx

# Sense the following requests sent to a list:
sense_requests '(^[     ]*Q?U?I?E?T?[     ]*ADD[      ]+([A-Z0-9.@-]+)[
      ]+.+[       ]+.+)|&
(^[   ]*ALIA?S?[  ]+([A-Z0-9.@_-]+)[      ]+.+[       ]+.+)|&
(^[   ]*APPR?O?V?E?[    ]+([A-Z0-9.@_-]+)[      ]+.+[       ]+[0-9]+[
      ]*$)|&
```

```
(^[   ]*ARCH?I?V?E?[    ]+([A-Z0-9._-]+))|&
(^[   ]*CONF?I?G?U?R?A?T?I?O?N?[    ]+([A-Z0-9.@_-]+)[      ]+.+[       ]+)|&
(^[   ]*Q?U?I?E?T?[     ]*DELE?T?E?[      ]+([A-Z0-9.@_-]+)[      ]+.+[
      ]+.+)|&
(^[   ]*DISC?A?R?D?[    ]+([A-Z0-9.@_-]+)[      ]+.+[       ]+[0-9]+[
      ]*$)|&
(^[   ]*EDIT[     ]+([A-Z0-9.@_-]+)[      ]+.+[       ]+[A-Z]+[   ]*.*$)|&
(^[   ]*FREE[     ]+([A-Z0-9.@_-]+)[      ]*.*$)|&
(^[   ]*GET[      ]+[A-Z0-9/.-]+.+)|&
(^[   ]*HOLD[     ]+([A-Z0-9.@_-]+)[      ]*.*$)|&
(^[   ]*SENDM?E?[       ]+[A-Z0-9/.-]+.+)|&
(^[   ]*VIEW[     ]+[A-Z0-9/.-]+.+)|&
(^[   ]*HELP[     ]*[A-Z0-9]*$)|&
(^[   ]*IGNO?R?E?[      ]+([A-Z0-9.@_-]+)[      ]+.+[       ]+.+)|&
(^[   ]*INDEX[    ]*[A-Z0-9/.-]*.*)|&
(^[   ]*INFO?R?M?A?T?I?O?N?[  ]*([A-Z0-9.@_-]*)[      ]*$)|&
(^[   ]*INIT?I?A?L?I?Z?E?[    ]+([A-Z0-9.@_-]+)[      ]*.*$)|&
(^[   ]*NEW-L?I?S?T?[   ]+([A-Z0-9.@_-]+)[      ]*.*$)|&
(^[   ]*JOIN?[    ]+([A-Z0-9.@_-]+).+)|&
(^[   ]*LIST?S?[  ]*$)|&
(^[   ]*LOCK[     ]+([A-Z0-9.@_-]+)[      ]*.*$)|&
(^[   ]*PUT[      ]+([A-Z0-9.@_-]+)[      ]+.+)|&
(^[   ]*PURG?E?[  ]+([A-Z0-9.@_-]+))|&
(^[   ]*RECI?P?I?E?N?T?S?[    ]+([A-Z0-9.@_-]+)[      ]*.*$)|&
(^[   ]*REVI?E?W?[      ]+([A-Z0-9.@_-]+)[      ]*.*$)|&
(^[   ]*RELE?A?S?E?[    ]*$)|&
(^[   ]*VERS?I?O?N?[    ]*$)|&
(^[   ]*REPO?R?T?S?[    ]+([A-Z0-9.@_-]+)[      ]+.+$)|&
(^[   ]*RUN[      ]+([A-Z0-9.@_-]+))|&
(^[   ]*SEAR?C?H?[      ]+([A-Z0-9.@_-]+))|&
(^[   ]*Q?U?I?E?T?[     ]*SET[      ]+([A-Z0-9.@_-]+))|&
(^[   ]*QUER?Y?[  ]+([A-Z0-9.@_-]+))|&
(^[   ]*STAT?I?S?T?I?C?S?[    ]+([A-Z0-9.@_-]+))|&
(^[   ]*SYST?E?M?[      ]+([A-Z0-9.@_-]+)[      ]+.+[       ]+#.+)|&
(^[   ]*SUBS?C?R?I?B?E?[      ]*([A-Z0-9.@_-]*))|&
(^[   ]*SIGN?O?F?F?[    ]*([A-Z0-9.@_-]*))|&
(^[   ]*UNLOCK[   ]+([A-Z0-9.@_-]+)[      ]*.*$)|&
(^[   ]*UNSUBS?C?R?I?B?E?[    ]*([A-Z0-9.@_-]*))|&
(^[   ]*SHUTDOWN[       ]+.+$)|&
(^[   ]*RESTART[  ]+.+$)|&
(^[   ]*EXEC?U?T?E?[    ]+.+#.+)|&
(^[   ]*WHIC?H?[  ]*$)|&
(^[   ]*REMOVE[   ]+M[EY])|&
(^[   ]*DELETE[   ]+M[EY])|&
(^[   ]*ADD[      ]+M[EY])|&
(^[   ]*PLEASE[   ]+REMOVE[   ]+)|&
(^[   ]*PLEASE[   ]+DELETE[   ]+)|&
(^[   ]*PLEASE[   ]+U?N?SUBS?C?R?I?B?E?[  ]+)|&
(^[   ]*PLEASE[   ]+SIGN[     ]+)|&
(^[   ]*PLEASE[   ]+ADD[      ]+)|&
(^[   ]*CONFIRMATION[   ]+OF[       ]+READING)|&
(^[   ]*THE[      ]+MESSAGE[  ]+.+SENT[   ]+WAS[      ]+DELIVERED)|&
(^[   ]*CONFIRMATION.+MESSAGE.+READ)'
```

```
# If using the "fax" request, tell the system where to find the fax program:
# fax /bin/fax

# Define UNIX commands for list test for "run" requests.
# The lists they refer to have to be defined beforehand, so make sure they
# are placed in the proper place:
#unix_cmd test pass1 dir '/bin/ls $*' #Syntax: dir [args]
#unix_cmd test pass1 swap '/bin/echo $2 $1' #Syntax: swap <arg1> <arg2>
#unix_cmd test pass1 who '/bin/who' #Syntax: who

# Define header lines to be preserved or discarded, globally, i.e.
# for all lists and listproc. Lines that start with ! are to be discarded:
header * {
  !Registered-Mail-Reply-Requested-By:
  !X-Confirm-Reading-To:
  !X-Phone:
  Pmrc:
}

# Tell the server which header lines to preserve from the sender's message
# Can use regular expressions (.+ preserves unknown headers):
# header test {
#   PEM-.+:
#   .+:
# }

# --- LISTS ARE DEFINED BELOW --- DO NOT REMOVE THIS LINE ---
# Define local lists below (automatically done by the system -- no user
# intervention required) and optional command line arguments:
# list test test@cren.net     # -D -e
```

17. ListProc Files

LPDIR/.aliases
ListProc global alias file. Addresses matching the regular expressions on the left are transformed according to the rules on the right.

LPDIR/.awk
awk program for formatting the STATISTICS request.

LPDIR/.grep
script used for counting the number of messages sent by a subscriber.

LPDIR/.ignored
Global list of addresses to ignore.

LPDIR/.lock.serverd
Lock file for serverd(1).

LPDIR/.message.ids
A database of recent message id's used to detect mail loops.

LPDIR/.queue.id
Next id to be assigned to next undeliverable message placed in the mqueue/ directory.

LPDIR/.reply.listser
The reply code given to serverd(1) during a live session.

LPDIR/.rep.server.acc
Archived listproc(1) reports. This is used only if **syslog** logging is not enabled.

LPDIR/.rep.serverd.acc
Archived serverd(1) reports. This is used only if **syslog** logging is not enabled.

LPDIR/.rep.start.acc
Archived start(1) reports. This is used only if **syslog** logging is not enabled.

LPDIR/.report.catmail
Current catmail(1) report. This is used only if **syslog** logging is not enabled.

LPDIR/.report.daemon
Current serverd(1) report. This is used only if **syslog** logging is not enabled.

LPDIR/.report.list
Error messages from list(1). This is used only if **syslog** logging is not enabled.

LPDIR/.report.server
Current listproc(1) report. This is used only if **syslog** logging is not enabled.

LPDIR/.report.start
Last start(1) action (system started or shut down). This is used only if **syslog** logging is not enabled.

LPDIR/.stats
Script used the information used by the STATISTICS request.

LPDIR/.sums
Database of checksums from the 2000 most recent error messages sent to the ListProc server. Note that the checksums of syntactically correct messages are not saved here, as that would prevent a user from issuing the same command more than once.

LPDIR/.tag.id
Next id to be assigned to next moderated message.

LPDIR/.warning
Log of system calls' exit statuses.

LPDIR/batch
Batched user requests.

LPDIR/catmail
Utility to append incoming messages to the proper incoming folder.

LPDIR/config
The main configuration file.

LPDIR/farch
File archiving utility.

LPDIR/ilp
Interactive ListProcessor client; to be used to connect to the interactive server.

LPDIR/list
The mailing list program.

LPDIR/listproc
Process incoming server requests.

LPDIR/lost+found
Requests that could not be appended to

LPDIR/requests
Incoming server request folder, in UNIX mbox format.

LPDIR/mbox
Archive of processed server requests, stored in UNIX mbox format.

LPDIR/news
Script used to add a news group to a list.

LPDIR/owners
List of list owners' addresses and Ccoptions.

LPDIR/peer
Script used to add a peer list.

LPDIR/pqueue
The mail queue processing program (see queue(1)).

LPDIR/priv.hosts
List of hosts that may connect to the interactive part of the system.

LPDIR/queued
The mail queue processing daemon (see queue(1)).

LPDIR/received
File containing the messages received from sendmail during the last message distribution. This is only used if the **mailmethod** is set to **system**. (See page 28 for information on setting your **mailmethod**.)

LPDIR/requests
Newly arrived user requests.

LPDIR/requests.live
Repository for requests coming in from live connections.

LPDIR/sent
File containing the messages sent to sendmail during the last message distribution. This is only used if the **mailmethod** is set to **system**. (See page 28 for information on setting your **mailmethod**.)

LPDIR/serverd
The system's daemon.

LPDIR/setup
System setup script.

LPDIR/start
The system's start program (see the description on page 83).

LPDIR/tlock
Script to checks for any locked files.

LPDIR/ulock
Script to remove all lock files.

LPDIR/unwanted.hosts
List of hosts that may not connect to the interactive part of the system.

LPDIR/welcome.live
Welcoming message upon live connection establishment.

LPDIR/archives/listproc/DIR
Directory of all files available from the master archive (listproc).

LPDIR/archives/listproc/INDEX
The master archive index.

LPDIR/doc/catmail.1
Man page source for catmail(1).

LPDIR/doc/catmail.nr
Formatted man page for catmail(1).

LPDIR/doc/farch.1
Man page source for the farch(1) utility.

LPDIR/doc/farch.nr
Formatted man page for the farch(1) utility.

LPDIR/doc/ilp.1
Man page source for the ilp(1) utility.

LPDIR/doc/ilp.nr
Formatted man page for ilp(1).

LPDIR/doc/list.1
Man page source for list(1).

LPDIR/doc/list.nr
Formatted man page for list(1).

LPDIR/doc/queue.1
Man page source for pqueue and queued(1).

LPDIR/doc/queue.nr
Formatted man page for pqueue and queued(1).

LPDIR/doc/serverd.1
Man page source for serverd(1).

LPDIR/doc/serverd.nr
Formatted man page for serverd(1).

LPDIR/doc/start.1
Man page source for start(1).

LPDIR/doc/start.nr
Formatted man page for start(1).

LPDIR/help/TOPICS
This defines the paths to the various help topics available on your server. If the paths in this file are incorrect, ListProc won't be able to find the help topics.

LPDIR/help/*
General and topic-specific help files available upon a help request

LPDIR/lists/LISTNAME/.aliases
Aliases of email addresses of subscribers.

LPDIR/lists/LISTNAME/.digest.toc
All the subject lines from the digest. It will be written at the head of the digest when it is sent.

LPDIR/lists/LISTNAME/.digest.msg
All the messages of the digest, separated by dashed lines.

LPDIR/lists/LISTNAME/.digest.time
The time that the last digest was sent, represented as the value returned by time(2). It is used to calculate when it is time to send out the next digest.

LPDIR/lists/LISTNAME/.headers
An archive of who sent each message; used upon a statistics request.

LPDIR/lists/LISTNAME/.ignored
A list of unwanted senders.

LPDIR/lists/LISTNAME/.info
Text sent upon an information request.

LPDIR/lists/LISTNAME/.limits
Used by serverd(1) when placing various limits for the list.

LPDIR/lists/LISTNAME/.message.ids
A database of recent message id's used to detect mail loops.

LPDIR/lists/LISTNAME/.news
A list of all news groups connected to this list.

LPDIR/lists/LISTNAME/.peers
A list of all peer lists for this mailing list, along with the email addresses of their servers.

LPDIR/lists/LISTNAME/.rep.list.acc
Archived reports from list(1).

LPDIR/lists/LISTNAME/.report.list
Current report from list(1).

LPDIR/lists/LISTNAME/.restricted
List of subscribers with alternate recipient files.

LPDIR/lists/LISTNAME/.subscribers
A list of all subscribers along with preferences and settings.

LPDIR/lists/LISTNAME/.un.digest
When a digest is constructed, it will be stored in this file until it has been sent to all digest subscribers.

LPDIR/lists/LISTNAME/.time.subscrib
This file is used to record the time of the last modification of the subscribers file.

LPDIR/lists/LISTNAME/.welcome
Text sent on a subscribe request.

LPDIR/lists/LISTNAME/lost+found
Mail that could not be appended to the 'mail' or 'moderated' files (see next entries and catmail(1)).

LPDIR/lists/LISTNAME/mail
Newly arrived public messages to be distributed.

LPDIR/lists/LISTNAME/mbox
An archive of all messages sent to date.

LPDIR/lists/LISTNAME/moderated
Newly arrived public messages that need to be edited before distributed.

LPDIR/lists/LISTNAME/removed.users
Entries from the .subscribers file for users who have been automatically removed from the list.

LPDIR/lists/LISTNAME/removed.alias
Entries from the .aliases file for users who have been automatically removed from the list.

LPDIR/mqueue/*
Queued messages for later delivery.

18. Regular Expressions

Many ListProc command lines take regular expressions as arguments. A regular expression is a group of symbols which describe a unique string of characters. An example of a simple regular expression is the word “donkey”. In a group of words, the regular expression “donkey” matches only other instances of the word “donkey” and nothing else. So if we had a text file and did a search for “donkey” every time that word appeared in the text it would show up in our search. This definition can be expanded if we define the period “.” as a wild card replacement for a single character. Then the regular expression “.onkey” would include “honkey” or “tonkey” or “bonkey” as well as “donkey”. We can continue to expand on the definition by adding the asterisk to the period “ .* ” as a substitute for any number of characters. Then the regular expression “don.* ” will not only include “donkey” but will include all strings of characters beginning with “don” including “don” itself. A search of a text file for “don.* ” will turn up dongle, donkey, donner, dondoodit, dondiddle, etc. Regular expression matching, therefore, puts together a whole set of rules that allow you to test whether a string fits into a specific syntactic shape. You can also search a string for a substring that fits a pattern, and just as importantly, you can replace one string with another. The command line of a ListProc command is more like a regular expression as used in a database search.

Regular expressions have a syntax in which a few characters are special constructs and the rest are “ordinary”. An ordinary character is a simple regular expression which matches that character and nothing else. The special characters are `$’, `^’, `.’, `*’, `+’, `?’, `[‘, `]’ and `\’. Any other character appearing in a regular expression is ordinary, unless a `\’ precedes it.

For example, ‘f’ is not a special character, so it is ordinary, and therefore ‘f’ is a regular expression that matches the string ‘f’ and no other string. (It does *not* match the string ‘ff’.) Likewise, ‘o’ is a regular expression that matches only ‘o’.

Any two regular expressions A and B can be concatenated. The result is a regular expression which matches a string if A matches some amount of the beginning of that string and B matches the rest of the string.

As a simple example, we can concatenate the regular expressions ‘f’ and ‘o’ to get the regular expression ‘fo’, which matches only the string ‘fo’. Still trivial.

The following are the characters and character sequences which have special meaning within regular expressions. Any character not mentioned here is not special; it stands for exactly itself for the purposes of searching and matching.

‘.’
is a special character that matches anything except a newline. Using concatenation, we can make regular expressions like ‘a.b’ which matches any three-character string which begins with ‘a’ and ends with ‘b’.

‘*’
is not a construct by itself; it is a suffix, which means the preceding regular expression is to be repeated as many times as possible. In ‘fo*’, the ‘*’ applies to the ‘o’, so ‘fo*’

matches ‘f’ followed by any number of ‘o”s. The case of zero ‘o”s is allowed: ‘fo*’ does match ‘f’.

‘*’
always applies to the *smallest* possible preceding expression. Thus, ‘fo*’ has a repeating ‘o’, not a repeating ‘fo’. The matcher processes a ‘*’ construct by matching, immediately, as many repetitions as can be found. Then it continues with the rest of the pattern. If that fails, backtracking occurs, discarding some of the matches of the ‘*”d construct in case that makes it possible to match the rest of the pattern. For example, matching ‘c[ad]*ar’ against the string ‘caddaar’, the ‘[ad]*’ first matches ‘addaa’, but this does not allow the next ‘a’ in the pattern to match. So the last of the matches of ‘[ad]’ is undone and the following ‘a’ is tried again. Now it succeeds.

‘+’
is like ‘*’ except that at least one match for the preceding pattern is required for ‘+’. Thus, ‘c[ad]+r’ does not match ‘cr’ but does match anything else that ‘c[ad]*r’ would match.

‘?’
‘?’ is like ‘*’ except that it allows either zero or one match for the preceding pattern. Thus, ‘c[ad]?r’ matches ‘cr’ or ‘car’ or ‘cdr’, and nothing else.

‘[...]’
‘[‘ begins a “character set”, which is terminated by a ‘]’. In the simplest case, the characters between the two form the set. Thus, ‘[ad]’ matches either ‘a’ or ‘d’, and ‘[ad]*’ matches any string of ‘a”s and ‘d”s (including the empty string), from which it follows that ‘c[ad]*r’ matches ‘car’, etc.

Character ranges can also be included in a character set, by writing two characters with a ‘-’ between them. Thus, ‘[a-z]’ matches any lower-case letter. Ranges may be intermixed freely with individual characters, as in ‘[a-z$%.]’, which matches any lower case letter or ‘$’, ‘%’ or period.

Note that the usual special characters are not special any more inside a character set. A completely different set of special characters exists inside character sets: ‘]’, ‘-’ and ‘^’. To include a ‘]’ in a character set, you must make it the first character. For example, ‘[]a]’ matches ‘]’ or ‘a’. To include a ‘-’, you must use it in a context where it cannot possibly indicate a range: that is, as the first character, or immediately after a range.

‘[^ ...]’
‘[^’ begins a “complement character set”, which matches any character except the ones specified. Thus, ‘[^a-z0-9A-Z]’ matches all characters *except* letters and digits. Note that the ^ has to be within brackets. Outside of brackets it has a different meaning as mentioned below. ‘^’ is not special in a character set unless it is the first character. The character following the ‘^’ is treated as if it were first (it may be a ‘-’ or a ‘]’).

‘^’
is a special character that matches the empty string — but only if at the beginning of a line in the text being matched. Otherwise it fails to match anything. Thus, ‘^foo’ matches a ‘foo’ which occurs at the beginning of a line.

‘$’
is similar to ‘^’ but matches only at the end of a line. Thus, ‘xx*$’ matches a string of one

or more ‘x”s at the end of a line.

‘\’
has two functions: it quotes the above special characters (including ‘\’), and it introduces additional special constructs. If you wanted to search for the dollar sign within a text you would have to use \$ in place of just $ since the dollar sign has a special meaning. Because ‘\’ quotes special characters, ‘\$’ is a regular expression which matches only ‘$’, and ‘\[‘ is a regular expression which matches only ‘[‘, and so on.

For the most part, ‘\’ followed by any character matches only that character. However, there are several exceptions: characters which, when preceded by ‘\’, are special constructs. Such characters are always ordinary when encountered on their own.

No new special characters will ever be defined. All extensions to the regular expression syntax are made by defining new two-character constructs that begin with ‘\’.

‘\|’
specifies an alternative. Two regular expressions A and B with ‘\|’ in between form an expression that matches anything that either A or B will match. Thus, ‘foo\|bar’ matches either ‘foo’ or ‘bar’ but no other string. ‘\|’ applies to the largest possible surrounding expressions. Only a surrounding ‘\(... \)’ grouping can limit the grouping power of ‘\|’. Full backtracking capability exists when multiple ‘\|”s are used.

‘\(... \)’
is a grouping construct that serves three purposes:

1. To enclose a set of ‘\|’ alternatives for other operations. Thus, ‘\(foo\|bar\)x’ matches either ‘foox’ or ‘barx’.

2. To enclose a complicated expression for the postfix ‘*’ to operate on. Thus, ‘ba\(na\)*’ matches ‘bananana’, etc., with any (zero or more) number of ‘na”s.

3. To mark a matched substring for future reference. This last application is not a consequence of the idea of a parenthetical grouping; it is a separate feature which happens to be assigned as a second meaning to the same ‘\(... \)’ construct because there is no conflict in practice between the two meanings. Here is an explanation of this feature:

‘\DIGIT’
After the end of a ‘\(... \)’ construct, the matcher remembers the beginning and end of the text matched by that construct. Then, later on in the regular expression, you can use ‘\’ followed by DIGIT to mean “match the same text matched the DIGIT’th time by the ‘\(... \)’ construct.” The ‘\(... \)’ constructs are numbered in order of commencement in the regexp.

The strings matching the first nine ‘\(... \)’ constructs appearing in a regular expression are assigned numbers 1 through 9 in order of their beginnings. ‘\1’ through ‘\9’ may be used to refer to the text matched by the corresponding ‘\(... \)’ construct.

For example, ‘\(.*\)\1’ matches any string that is composed of two identical halves. The ‘\(.*\)’ matches the first half, which may be anything, but the ‘\1’ that follows must match the same exact text.

‘\b’
matches the empty string, but only if it is at the beginning or end of a word. Thus, ‘\bfoo\b’ matches any occurrence of ‘foo’ as a separate word. ‘\bball\(s\|\)\b’ matches ‘ball’ or ‘balls’ as a separate word.

‘\B’
matches the empty string, provided it is *not* at the beginning or end of a word.

‘\<‘
matches the empty string, but only if it is at the beginning of a word.

‘\>’
matches the empty string, but only if it is at the end of a word.

‘\w’
matches any word-constituent character.

‘\W’
matches any character that is not a word-constituent. What the ‘\(...\)’ groupings matched.

Here are examples of commands that use regular expressions:

```
lists global ‘health|mental’~death
```

The above will compile a list of lists that contain either the word ‘health’ or ‘mental’ in either their list name or description comment but will exclude lists with the word ‘death’. The way you should read ‘health|mental’~death out loud is; “health or mental but not death”.

```
lists global move$&dan$
```

will search for all lists containing BOTH the characters ‘move’ AND ‘dan’ so that move$ will return both movement and movies and dan$ will return both dancing and danger. But in order for you to receive a reply, the list will have to contain BOTH words. So a list about Dangerous Movies will show up in your search as well as a list about Movement and Dancing.

```
ignore <listname> <password> bart@^ar..+beta.org
```

This example will ignore a user named bart whether he posts from ar1.beta.org or from ar2.beta.org or art.beta.org.

```
alias <listname> <password> (.+)@host.domain \1@domain
```

will take anything inside the () and store it in \1 which is then accessed. For example, if homer@cs.domain sends a message, it will be translated to homer@domain.