ListProcessor®

ver 7.1
4 January, 1995

User Manual©

The Corporation for Research and Educational Networking

ListProcessor®
User Manual©
Table of Contents

ListProcessor®
User Manual©

I. Introduction

This User Manual provides a complete introduction to the CREN® ListProcessor® list- and file-management software, also known as ListProc®. It should explain to you in detail what ListProcessor is, how to use it, and give you an introduction to the accepted standards of conduct on the Internet, commonly known as "Netiquiette".

II. What is ListProcessor?

ListProcessor (ListProc, for short) is a powerful mailing list agent that keeps track of thousands of people subscribed to any number of mailing lists. Users may subscribe to lists, post messages, review members of lists, review the configuration set up of lists, etc. It allows full control over the list and its configuration to the list owner, freeing up the system manager from having to make major modifications for the list owners when needed. ListProc is also a powerful file archiver, allowing search and retrieval of text files based on regular expressions and e-mail retrieval of binary files automatically uuencoded and divided into managable sized portions. It is automated, and eliminates the need for user intervention and maintenance of multiple aliases of the form "list, list-owner, list-request", etc. There is support provided for public and private hierarchical archives, moderated and non-moderated lists, peer lists, peer servers, private lists, address aliasing, news connections and gateways, mail queueing, digests, list ownership, owner preferences, crash recovery, batch processing, configurable headers, regular expressions, archive searching, and live user connections via TCP/IP.

III. ListProcessor Commands

Your interaction with ListProc will generally be by sending e-mail messages to ListProc or by sending e-mail messages to a discussion group, or "list", to which you are subscribed. There is also a mechanism of interactively sending commands to ListProc using a program called Interactive ListProcessor, or "ilp" which will be described in further detail below.

Note that all commands described below <u>must</u> be sent to a ListProcessor server for processing. A ListProcessor server will have an address that is in the form of *listproc@host.domain*, such as *listproc@listproc.net*. Commands must <u>not</u> be sent to a mailing list, since all mail to a mailing list is sent to all the subscribers to that list.

Sometimes you need to contact the moderator of an Internet mailing list rather than post to the entire list. To do this you should know that there are, by convention, two mailing addresses for every mailing list (except where noted by the List of Lists):

list-request@host   (e.g. xpert-request@x.org)
list@host   (e.g. xpert@x.org)

Any e-mail sent to list-request@host, where “list” is replaced by the name of the list and “host” is replaced by the host machine name, will either be sent to the owners of the list rather than to the list or return information about the owner(s) of the list. Mail sent to list@host will be sent out to everyone subscribed to the list. Examples are given in parenthesis above. If you are subscribed to a list named “xpert” whose host machine is x.org and want to send a message to the owners, you would send that message to xpert-request@x.org.

The following syntax is used in this document: ListProcessor commands are shown in a **`different boldface typeface`** from the text, with the minimum abbreviation in **`UPPERCASE`** letters. All commands can be abbreviated and require that you use a sufficient number of characters in the abbreviation to distinguish the command from any other command, with a minimum of three letters. In most cases the minimium abbreviation usable for each command is the first three letters of that command. The actual commands are case-insensitive. For example, the command **`SUB`** is the same as **`subscribe`** so in the text below it is written as **`SUBscribe`** to let you know that you can abbreviate the command as **`SUB`**. Command arguments and options in which the term used is to be replaced by the user are shown in ***italics***. Optional arguments and keywords are enclosed in brackets (**`[...]`**), which must be omitted when the actual command is used. Alternatives are separated by a vertical bar (**|**). A list *explore@listproc.net* has been established for anyone to use to explore and experiment with ListProc.

"Local" lists are those hosted by the ListProcessor with which you are communicating; "remote" lists are those known by that ListProcessor to be hosted by other ListProcessors. Commands which relate to remote lists are forwarded to the appropriate ListProcessor whenever possible.

For additional information, including specialized commands, use the **`HELP`** command described below or refer to the Listprocessor User Reference, available in plain text format by sending the command **`get doc ref`** as the body of a mail message to *listproc@listproc.net* or any other CREN ListProcessor. This card is available from *info.cren.net* via anonymous ftp, as the PostScript file */listproc/ps/doc-card.ps* and as the RTF file */listproc/rtf/doc-card.rtf* (open the latter with Microsoft Word or WordPerfect®, and set the orientation to landscape before printing).

IV. General ListProcessor Commands

`HELp`

An e-mail message to ListProc, at the host address of the list you are interested in, containing only the word **`HELP,`** will return to you an e-mail message containing a brief description of every ListProcessor command (when no topic is specified). Your message should contain only the word **HELP** and nothing else. It should ***not*** say things like “help me” or “Please send help” because ListProc automatically generates the response to your request for help and is not capable of actually reading messages. ListProc will send the response back you and you do not need to add your return address or a signature to the message.

HELp TOPICS

There are several individual topics that ListProc can send help about, such as individual commands listed below. A message to ListProc saying **HELP TOPICS** returns a list of all the topics for which help is available.

HELp *topic*

After you have a list of topics for which ListProc can send you additional information you can get that information by sending a message to ListProc saying **HELP** followed by the one word topic you want information on. This returns information about the topic selected, if available.

HELp LISTProc

returns information about ListProcessor and other list-management systems.

HELp LIVe

returns information about accessing ListProcessor interactively.

SUBscribe *list-name your-name*

UNSubscribe *list-name*

In order to be able to receive messages from the list you are interested in you must subscribe to that list. And in many cases you also must be subscribed to a list in order to send messages to other subscribers. Use the first of these commands to add yourself to the list whose ***list-name*** you specify and the second to remove yourself from the list whose ***list-name*** you specify (to which list you must, of course, have previously **subscribed**). In these commands, replace ***list-name*** with the name of the list (that part of the e-mail address of the list to the left of the @-sign). In the **subscribe** command, replace ***your-name*** with your real name (not your e-mail address). For example, if your name is Joan S. Doe and you wish to subscribe to the list *circuses@barnum.com*, you may join that list by sending mail with the command

subscribe circuses Joan S. Doe

The subscribe command can be abbreviated by using the first three letters of the command. This is true of all commands. Therefore you can also use the command

sub circuses Joan S. Doe

as the body (not the subject) of the message, to your local ListProc. If your local ListProc does not know about lists hosted by other ListProcessors, you will have to send the message to *listproc@barnum.com*. Notice that in this command you do not have to include your e-mail address, you only need to include your full name. This is because ListProc reads the return address on your message and uses that as your subscription address. This means that you cannot subscribe to a list by sending a subscribe command from your friend’s e-mail account. User names may not contain the following characters: `"<>[]{}|$~*?!\ These are illegal characters as far as listproc is concerned.

PURge *password*

If you are subscribed to several lists that operate off of one host and you want to remove yourself from all of them, this command will remove you from all local lists. Use a valid password from one of the lists to which you are subscribed.

WHIch
This command returns a message to you listing all local mailing lists to which you have subscribed.

LISts local
LISts global
Each machine that runs ListProc has a number of lists that it runs which are called “local lists” and is aware of lists, called “remote lists”, on other machines. The **LISTS** command returns to you a list of all lists the Listprocessor knows about. **LISTS LOCAL** returns a message listing all local mailing lists with a one line description of each list. **LISTS GLOBAL** returns a message listing all local and remote lists known to the server with a one line description of each list. You MUST specify *local* or *global* in a **LISTS** command.
LISts local|global [keywords]
Adding a keyword to the **LISTS** command causes the ListProcessor to search only for those lists containing that keyword in its list name or one line description. ListProc will accept a *regular expression* as a keyword. Multiple keywords are treated as a logically ANDed list of strings/*regular expressions*, i.e. only lists containing ALL the keywords will be returned. Keywords may be surrounded by quotes. For a discussion of regular expressions, see the section on that subject below.

REView *list-name* [SHOrt|DEScription|SUBscribers]
Every list has attributes associated with it which determine how the list operates and what privileges you have in sending commands to the ListProcessor concerning the list. The **REView** command allows you to get the list's attributes or settings and a list of all the people subscribed to the list. In some cases the list owner may limit your ability to obtain any or all of the information; in that case you will get back a message saying what information is denied you.
REView *list-name*
will send you a short file of general information about the list and a listing of the current unconcealed subscribers for the specified list. If there are a lot of subscribers to the list this message can be very long.
REView *list-name* SHOrt
REView *list-name* DEScription
will result in your receiving only the short description file, not the subscribers.
REView *list-name* SUBscribers
allows you to get only the non-concealed subscribers list. If a list is private, members only may issue this command. If the list is linked with any peer lists, your command will be forwarded to the appropriate server(s) also (except when using *ilp*).

For example, if you wanted to know about the settings of the list bajor-l but did not want to see the subscribers, you would send the command **`review bajor-l short`**.
`review bajor-l short` returns the following response:

```
***
*** list bajor-l@somewhere.net: Ongoing discussion of politics on planet Bajor
***
***  Date created: Sun Sep 11 03:39:41 1994

--- The current list settings are as follows:

PRIVATE: subscriptions controlled by jim@machine.org sally@machine.org .
SEND: open to subscribers and owners only.
VISIBLE: the list shows up in listings.
NO-ARCHIVE: no logs are kept.
STATS: open to subscribers and owners only.
REVIEW: open to subscribers and owners only.
ARCHIVES: available to subscribers and owners only.
UNMODERATED: postings not controlled.
DIGEST: digests distributed daily at 00:01
MESSAGE-LIMIT: unlimited daily postings.
FORWARD-REJECTS: no; all listproc-generated errors sent to sender.
REPLY-TO-LIST
AUTO-DELETE-SUBSCRIBERS: yes.
KEEP-RESENT-LINES: yes; Resent- header lines preserved.
DELIVERY-ERRORS: non-delivery reports are sent to the owners.
OWNERS: jim@machine.org sally@machine.org


bajor-l is a private, closed list used by the Bajoran council to discuss local politics.  Minutes of the
meetings of the council are posted to the archives.
```

`SET` ***`list-name`*** **`[`*`option arg`*`[`*`s`*`]]`**
Every user has user settings affiliated with his/her subscription to a list. The **`SET`** command is used to change your settings to whatever is convinent for you. If you send the command **`SET`** ***`list-name`*** without the optional arguments, ListProc will send back a list of all your current settings for the specified list as in the following example.

```
Current settings are:

ADDRESS = ODO@MACHINE.ADDRESS.ORG
MAIL = NOACK
PASSWORD = 749472019
CONCEAL = NO
```

The options available with the **`SET`** comand are **`mail`**, **`password`**, **`address`**, and **`conceal`**. Each option is further explained below:

`SET` ***`list-name`*** **`mail`** ***`arg`***
Arg may be one of: **`ack|noack|postpone|digest`**
`ack|noack`: A list owner can set up the list such that ListProc either *does* or *does not* send you a copy of all mail you send to a list. The **`ack`** and **`noack`** commands allow you to chose whether you want to receive a copy of any mail you post to a list. The major reason people would want to receive a copy of their own message

from the list is to confirm or acknowledge that the message actually went out. Thus, if you were subscribed to a list named aleph, **`Set aleph mail ack`** causes you to receive an acknowledgement copy of mail you send to the list *aleph*. **`Set list-name mail postpone`** causes ListProc to hold your mail until released by another **`set list-name mail`** command. This is useful if you go on vacation and don't want your mailer to fill up with messages while you are gone. Perhaps the system where you get your mail has a limit on the number of messages it will store for you and will reject anything over such a number. You can set your mail to postpone and then when you are ready to receive your mail again, you send another mail command such as **`set list-name mail ack`** and you will receive all the mail that was held for you.

It is quite possible that you have subscribed to a very active list which sends out a number of messages daily. If you would like to obtain all the messages contacenated together into a single digest of messages once a day you would send the command **`set list-name mail digest`**. This command would cause ListProc to send you all messages as a digest at the time and rate specified by the list owner. The list owner can specify that digests get sent out when their size reaches a specified amount or at a specific period of time.
For all four options of the set listname mail command, ack, noack, postpone, or digest, you can choose only one option to be active at any one time. Changing your mail option negates the previous setting. Therefore you cannot have your mail set to ack and digest at the same time or digest and postpone at the same time. It would not make any sense anyway since you cannot postpone receiving mail and at the same time receive a digest, or ask for a copy of your mail in an ack command when you will receive it back anyway in a digest. Likewise, if you want to turn off one of the options, for example if you want to turn off digest, you simply send another **`set list-name mail`** command You may also set your mailmode to the default for your list by sending the command **SET *list-name* mail.**

SET *list-name* password *old-password new-password*
Changes your list password. Your initial password, if needed, is supplied in the message you receive confirming your subscription.

SET *list-name* address *password new-address*
Change your registered e-mail address. In the event you suddenly lose your e-mail account and can no longer access it, you can tell ListProc to change the address by which you receive mail with this command. Since you are likely sending this command from an account other than the one you subscribed from, this command requires that you use your list password that you received in the message confirming your subscription.

SET *list-name* conceal yes|no
In some cases a person who is subscribed to a list does not want other people to have access to the fact that he/she is subscribed. If you set conceal to **YES,** `then the command to` **REView *list-name* subscribers** will *not* show your name. If you set conceal to **NO** then anyone will be able to see that you are subscribed to

the list. You may set yourself back to the original list default by sending the command
SET *list-name* conceal with no arguments after.

V. Archive commands (GET, INDEX, SEARCH)

Many lists maintain archives. Archives may contain copies of all old mail that has passed out over the list or may contain special files that the list owner didn't want to send out via e-mail but did want to make available to all subscribers. Archives may be public or private. Private archives require the /***password*** parameter. In the following, ***archive*** may be replaced by:

list-name At most sites, the list-name and archive are the same. If your site does not recognize your list-name as a valid archive, check with your list owner.

archive-name An alternate name given to the archive by the list owner.

path-to-archive – the full path to the archive as returned by **INDEX**.

INDex *archive* [/*password*] [-all]
List files in the selected archive, or the master archive if no archive was specified. The path to the archive is returned with the index. If you send an **INDEX** command without the name of the list or archive, ListProc will return an index of all public archives available on the host system. Archives may contain multiple subarchives; using a directory tree as an example, an archive may consist of a root archive with multiple subdirectories or subarchives within. The **INDEX** command will return only the one level of archive or directory requested; in order to obtain the entire archive structure including all subarchives the **-all** arguement must be added onto the **INDEX** command. In the event the archive is private you will have to use your list password or a password given to you by the list owner in order to obtain an index. The password must be preceded by a slash "/" when including it in an **INDEX** command.

SEArch [*archive*] [/*password*] [-all] [*pattern*]
If you are looking for a specific file but do not know what archive it is located in, you can search all files of the host system or of a specific archive (and all of its subarchives if **-all** is specified) for lines that match ***pattern***. Once again, if the archive is password protected you must specify the password with a slash "/" before it.
[Note: ***Pattern*** may be enclosed in single or double quotes and can be a *regular expression* with support for these additional operators:

'**^**' provides negation
'**|**' and '**&**' provide logical OR and AND
'**<**' and '**>**' are used to group parts of regular expressions
'**.**' matches any character including new line

See the discussion on regular expressions below.

GET [*archive*] file [/*password*]
Get ***file*** from the specified archive. Once you have located the file you want to get using either an **INDEX** or a **SEARCH** command you can get that file with a get

command. The requested file will be e-mailed to you. If the file is very large it may be split into multiple smaller parts in order to be e-mailed to you. Binary files cannot be sent via e-mail so if the file is a binary file it will be encoded to text using uuencode and you will have to obtain a copy of uudecode in order to convert the file back into it's original binary type. In the event the file is located in a private archive you will have to use your list password or a password given to you by the list owner in order to obtain the file. The password must be preceded by a slash "/" when including it in a **GET** command.

VI. Subscribing to Archives

AFD, Automatic File Distribution, is a mechanism for the automatic distribution of file archives or individual files via listproc. Users subscribe to an archive just as they would a message-based list. When the archive manager updates the files in the archive, the files are automatically sent out to the archive subscribers, or, if subscribers prefer, they may simply receive a notification of a file update, leaving it up to the subscriber to **GET** the file.

AFD <*action*> {<*archive*> [/*password*] [*files*]} & [{<*archive*> [/*password*] [*files*]}]

This will subscribe you to an archive such that whenever the archive is updated you will receive the new files automatically. You may optionally specify filenames in the archive and receive only those files when they are updates. The name of the archive and optional file names must be in brackets {}. If subscription to the archive requires a password then you put the password after the archive name with a slash "/" before it.

FUI <*action*> {<*archive*> [/*password*] [*files*]} & [{<*archive*> [/*password*] [*files*]}]

This will subscribe you to an archive such that whenever the archive is updated you will automatically receive notification of the new files but you will not receive the new files themselves as in the **afd** command. You may optionally specify filenames in the archive and receive only those files when they are updates. The name of the archive and optional file names must be in brackets {}. If subscription to the archive requires a password then you put the password after the archive name with a slash "/" before it.

AFD ADD will add your address to the specified file(s) of the specified archives notify lists, and when these files are updated a copy will be sent to you. You may add yourself to a full archive, instead of specified files, and you will be sent any updated files from that archive.
AFD DELETE will delete your address from the lists of the specified archives an/or files.
AFD QUERY will tell you which archives and/or files you have subscribed to.

FUI is similar to **AFD**, but instead you are notified when a file is updated, instead of being sent a new copy of the file. The **ADD**, **DELETE**, and **QUERY** commands for **FUI** have the same meanings as in the **AFD** command.

Examples:
For an archive called "bicycling" which requires a password of "bigwheels" and a second archive called "cooking":

afd add { bicycling /bigwheels ref.ps user.ps } { cooking }

This will add your address to archive cooking's notify list and you will be sent files (recipes) as updated, and to archive bicycling list, but only for files "ref.ps" and "user.ps" -- any other files updated will not be sent to you. This example assumes that bicycling is a private archive protected by password " bigwheels ".

afd delete { bicycling ref.ps }

This will remove your address from archive bicycling list for file ref.ps.

fui query { bicycling }

This will return you a list of files you are subscribed to in archive bicycling.

VII. Command Aliases

Many ListProcessor commands support aliases for compatibility with other list-management software. If you are familiar with some other list management software and cannot figure out how to send a similar command to ListProc, try the command your other software accepts. ListProc may be able to interpret it.

VIII. Interactive ListProcessor (ilp)

Commands to ListProcessor may be issued interactively, i.e., directly to ListProc without using e-mail, if both the host running ListProc and the system you use are networked directly on the Internet. ILP clients are available to run under UNIX, X-windows, Macintosh, and MS-Windows. An ilp session offers a command line interface in UNIX and graphical interface in the other clients. Copies of ilp should be available on the server hosting the list that you interact with. If not you may contact CREN for the latest version.

If you are using UNIX you will need to compile the ilp client before you use it. Then you connect to your desired server with the command "ilp host@domain" where "host@ domain" is replaced by the Internet domain name for the server to which you are connecting. It may be necessary to give an optional port address; check with your ListProc system manager if this is necessary. Once you connect you will be prompted for a username and password. Log in with the e-mail address that ListProc knows for you in your list as a login name and use your list password as a password. If you are not subscribed to any lists on the system you may login as "test" giving a password of "new-user". Once a connection has been established, you will get the ilp command prompt "REQUEST>" to which all commands are entered. In the examples below the command prompt is in regular type as above and the commands that you type are in **bold** type. If ListProc cannot match your e-mail address and password with one in a known list then ListProc will give you "casual user" privleges. Casual users may only issue commands for **help**, **information**, **recipients**, and **statistics** for public lists and may issue **index**, **get**, **view**, and **search** requests in the archives. If you are a subscriber to a list then you may issue all these plus the **set**, **run**, **subscribe**, **unsubscribe**, and **which** commands. List owners may

additionally issue all the owner commands for their lists. To review your command privleges type a question mark “**?**” or “**privleges**” at the prompt. ListProc accepts all ilp commands exactly as it does via e-mail. Long requests may be continued on multiple lines with an ampersand “&” at the end of each line. With the UNIX version of ilp the output of every request may be redirected to a file by using a standard UNIX redirection “>“ or “>>“ followed by a file name. For example, to get an index of an archive typing “REQUEST> **index > listproc.index**” will obtain an index of the archives and save it in a file called listproc.index. Like ftp, ilp allows you to specify a filetype for all files transfered using the keywords “binary” and “ascii”. Input may also be redirected from a file using the UNIX redirection “<“ followed by a filename. Therefore you can create a file called BATCH.REQUESTS containing the lines
index>index.text
get ListProc info >>index.text
quit
and run it with the command:
“REQUEST> **<BATCH>REQUESTS**”.
You can also use UNIX porting of output with the “|” such that the command:
“REQUESTS> **review listname | more**”
will pipe the output of the “review “ command to the UNIX “more” command. As a further example, you can look for a specific subscriber in a list by issuing the command:
“REQUESTS> **review listname | grep -i name**” where “name” is replaced by the person’s name.
To end an ilp session just enter quit or exit at the prompt.

IX. 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 characters and character sequences which have special meaning within regular expressions are referred to as “operators”. Some of the operators recognized by ListProcessor are listed below.

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.

search mylist-l bart@^ar..+beta.org

This example will search for messages in the mylist-l archive for all messages containing references to a user named bart whether he posts from ar1.beta.org or from ar2.beta.org or art.beta.org. In this manner you can turn up all his messages no matter which machine he posted from.

X. Netiquette, Good Manners on the Internet

Now that you have subscribed to all the lists that interest you, how are you going to interact with your newfound Internet community? Over the years users of the Internet have formed their own set of social norms, rules for interaction on the Internet, refered to as Netiquette.

Netiquette: The forms, manners, and actions established by the Internet community by convention as acceptable or required behavior in social interactions via e-mail. In other words, politeness to your fellow list subscribers.

- The rules of netiquette can be boiled down to one sentence; Never forget that the person on the other side is human. Because your interaction with the network is through a computer it is easy to forget that there are people "out there." Situations arise where emotions erupt into a verbal free-for-all that can lead to hurt feelings. Please remember that people all over the world are reading your words. Try not to say anything to others you would not say to them in person in a room full of people. Do not attack people if you cannot persuade them with your presentation of the facts. Screaming, cursing, and abusing others only serves to make people think less of you and less willing to help you when you need it. If you are upset at something or someone, wait until you have had a chance to calm down and think about it.

- Be brief. Never say in ten words what you can say in fewer. Say it succinctly and it will have a greater impact. A good message is only one or two screenfuls in length. Remember that the longer you make your message, the fewer people will bother to read it.

- Use descriptive subject lines. The subject line of a message is there to enable a person with a limited amount of time to decide whether or not to read your mesage. Tell people what the message is about before they read it. A title like "What the Hell" does not help as much as "My Messages to Listproc Are Bouncing".

- When you post a message to a list, think about the purpose of the list and the audience you are posting to. Asking auto repair questions on a list devoted to discussions of ancient Greece will not reach as many of the people you want to reach as if you asked them on a list devoted to auto repair. And it will certainly get most of the list's subscribers rather pissed off. It is considered bad form to post inappropriately to a list, especially if you are advertising something. Please do not use the Internet as an advertising medium. Advertisements on the Internet are rarely appreciated. Nothing will get you as many angry responses as advertising your commercial products on a discussion list. It is generally considered rude to post private e-mail correspondence without the permission of the author of that mail. Furthermore, under copyright statutes, the author of the e-mail possesses a copyright on mail that he or she wrote; posting it to the net or mailing it on to others without permission of the author is likely a violation of that copyright as well as being rude.

- "This is a Test: Don't Read This Message" If you want to try a test of something, do not use a world-wide list! Messages that say "This is a test" are likely to cause large numbers of caustic messages to flow into your mailbox. Listproc will automatically discard any message that has the word "Test" in the subject line.

- Be careful with humor and sarcasm. Without the voice inflections and body language of personal communications, it is easy for a remark meant to be funny to be misinterpreted. Subtle humor tends to get lost, so take steps to make sure that people realize you are trying to be funny. The net has developed a symbol called the smiley face. It looks like ":-)" and points out sections of messages with humorous intent (see section on smileys and abbreviations below). No matter how broad the humor or satire, it is safer to remind people that you are being funny. But also be aware that quite frequently satire is posted without any explicit indications. "Cute" misspellings are difficult to read, especially if the reader is not fluent in the language involved and even for people who are fluent in your language it can be quite tiresome to decipher.

- When replying to a message, summarize the part of the message to which you are replying. This allows readers to appreciate your comments rather than trying to remember what the original message said. But do not include the entire message since it will irritate the people who have already seen it.

- Be careful about copyrights and licenses. Once something is posted onto the network, it is *probably* in the public domain but many items have copyright notices and you should clear your actions with the author if you intend to redistribute such materials. Absolutely do not post copyrighted material without permission. Newspaper articles are copyrighted.

- Cite appropriate references. If you are using facts to support a cause, state where they came from. Don't take someone else's ideas and use them as your own.

- It is a good idea to have a few lines at the bottom of your message with your name and e-mail address because some mail readers do not display message headers and some novice users cannot decipher message headers. However, don't overdo signatures; keep them short. The main purpose of a signature is to help people locate you, not to tell your life story or show what a great artist you are.

- Try to keep your text in a generic format. Many, if not most, of the people reading your messages do so from 80 column terminals or from workstations with 80 column terminal windows. Try to keep your lines of text to less than 80 characters for optimal readability. If people quote part of your message in a followup, short lines will probably show up better, too. Also realize that there are many, many different forms of terminals in use. If you enter special control characters or tabs in your message, it may result in your message being unreadable on some terminal types.

- The Internet is notorious for use of abbreviations and symbols for conveying various feelings and other information. Smileys are one sort of universal symbol of the net. :-) This is the net convention for a "smiley face". It means that something is being said in jest or that the person is generally happy. If it doesn't look like a smiley face to you, flop your head over to the left and look again. A large number of variants exist and mean related things; for instance, :-(is sad. There are smiley dictionaries available on the net in various locations; ask around if you are interested. A few of the more commonly used abbreviations are BTW, WRT, FYI, IMHO, FQDN, RTFM, and FAQ. BTW is shorthand for "by the way." WRT is "With respect to". FYI is "For Your Information" and IMHO is "In My Humble Opinion" or "In My Honest Opinion." FQDN is a “fully-qualified domain name”, that is, a hostname containing full, dotted qualification of its name up to the root of the Internet domain naming system tree. For example, info.cren.net is a FQDM but cren is not. RTFM is generally used as an admonition and means "read the f*ing manual" (choice of f-words varies according to reader). The implication is that the answer to a query or complaint is easy to find if one looks in the appropriate location FIRST. Many lists have FAQ postings (Frequently-Asked Questions) to answer these questions.

- When you have something for everyone on the mailing list to read, mail to the list@host address. However, if you have an administrative question of the list owner, for example, "where are the archives?" or "what is this mailer error I got from sending to this list?" you send your message to list-request@host, which goes only to the mailing list administrator. On the other hand administrative commands such as requests to be added to a list or removed from a list should be addressed to listproc@host where your list resides and formulated in the proper format for a ListProc command as outlined in this manual. It is considered to be in bad taste to send administrative requests to the entire mailing list in question, and if (as is occasionally the case) the administrator does not read the mailing list (i.e. he just takes care of the admin tasks for the list), he will not see your request if you don't send it to the right address.

- Have you heard these stories about a dying child wanting postcards/get-well cards/business cards to get in the Guinness Book of World Records? Don't post it. Don't ask anyone to send him anything. He is now alive and well having had successful brain surgery and it's been almost 10 years now since his request. The story of the little boy keeps popping up, even though his mother and the agencies involved have been appealing for people to stop. So many postcards were sent that the agencies involved in the effort don't know what to do with them. The Guinness people have recorded the boy, Craig Shergold, as the record holder in the category. For confirmation, you can see page 24 of the 29 July 1990 NY Times or call the publisher of the Guinness Book. Craig Shergold (born 1979) of Carshalton, Surrey when undergoing cancer chemo-therapy was sent a record 33 million get-well cards until May 1991 when his mother pleaded for no more. A successful 5 hour operation on a brain tumour by neurosurgeon Neal Kassel at the University of Virginia, Charlottesville, USA in March 1991 greatly improved his condition. If you want to do something noble, please consider some worthwhile charity. There are tens of thousands of children dying around the world daily, and they could use more than a postcard.

- And another electronic chain letter also is making the rounds. The letter promises millions of dollars in just a few months. All you have to do is send a few dollars to someone listed in the message and add your name to the bottom of the message. Don't even think about it!!!! Trying to use the net to make vast sums of money or send chain letters is a very bad idea and in the US, you can (and will) be reported by pissed-off system administrators for fraud. Bottom line: don't try clever schemes to sell things, solicit donations, or run any kind of pyramid or Ponzi scheme. Also, don't start or support electronic chain letters.

- Be considerate with your use of network resources. Your individual usage may not seem like much compared to the net as a whole, but in aggregate, small savings in disk or CPU add up to a great deal.

XI. Your Feedback Desired

Your suggestions for improving this document are eagerly solicited by CREN and should be sent to *suggestions@listproc.net*.

® CREN is a registered Service Mark of the Corporation for Research and Educational Networking (CREN). CREN has applied for registration of ListProcessor and ListProc.