NAME

`sift-tcl` - Execute Sift-Tcl programs to sort e-mail messages

SYNOPSIS

`sift-tcl` [`-f` *`sift-tcl-file`*] [`-spool`] [`-log`] [`-v`]

DESCRIPTION

`Sift-tcl` is an extended Tcl interpreter for sifting (sorting, searching, routing) Internet e-mail. It is part of Sift-Mail and includes Tcl extensions for opening mail folders, parsing message headers and e-mail addresses, sending messages, and appending mail to folders. The Sift-Mail package also includes `sift-mail`, a graphical user interface which automatically generates Tcl code for `sift-tcl`, so non-expert users can easily set up the sifter as desired, for common tasks.

SETUP

`Sift-Tcl` is most commonly used to filter incoming mail. This is the mode best supported by `sift-mail`. To use `sift-tcl` in this fashion it must be executed when each new message arrives. To set this up, create a file called `.forward` in your home directory containing a line of the form:

```
"|/usr/local/bin/sift-tcl -spool"
```

Before you do this you should confirm that `sift-tcl` has been installed on your system in `/usr/local/bin`, and adjust the path name in the `.forward` file if it's not at that location. (You can sometimes find it by giving the command `which sift-tcl`.) You should also be sure that the path is absolute (it begins with a "/"). After you've created the `.forward` file send yourself a mail message to make sure mail is going through. If something has gone wrong with the set up, sifting might not work, or worse all your mail will be returned to the sender. `Sift-tcl` has a number of safety features to make sure that mail is delivered to your inbox even if errors occur. It also writes a log of actions and errors to the file `.siftlog` in your home directory.

Once it is installed, you can set up individual sifters for sifting tasks such as filtering mail messages from mailing lists by using `sift-mail`.

OPTIONS

`-log` Causes a log of actions and errors to be written to the log file `.siftlog` in your home directory. It also causes any automatic reporting specified in the Sift-Tcl configuration to be executed. All errors are written to the standard error as well as to the log.

`-spool` For use in a `.forward` file where `sift-tcl` is passed a message and must assume responsibility for its delivery. The `-spool` option will cause `sift-tcl` to attempt to read a message from the standard input and deliver it to the inbox if there are any errors executing the Tcl code. If this can't be done, it exits with a non-zero exit code to return the message to the delivery system. This also implies the `-log` option and that the Sift-Tcl code will be take from `.sift-tcl` in your home directory unless the `-f` option is given.

`-v` Causes a log of all actions to be written to the standard output. This is independent of the `-log` option.

`-f` *folder* Specify the Sift-Tcl code to execute. This must be given unless the `-spool` option is given. It can also be given with the `-spool` option to override the default file it selects.

OPERATION

`Sift-tcl` always executes the Tcl program specified by the `-f` option or takes the default, `.sift-tcl`, if the `-spool` option is given. The Sift-Tcl program can read messages from existing mail folders or from the standard input (see next section). It can move messages from one folder to another, send messages and change their status. The program is executed by a standard Tcl interpreter so all the standard Tcl features are available.

The `-spool` options enables additional error processing. This is intended for use when `sift-tcl` is invoked in a `.forward` file. If it fails to deliver the message properly or return it to the delivery system it will be lost. `Sift-Tcl` proceeds by attempting to save the message to the user's inbox. If that fails it exits with a non-zero exit code to signal the delivery system that delivery failed. The `-spool` option also turns on logging.

When either the `-spool` or `-log` option is given, and meta-information specifying that a sifting report be sent to the user is in the Sift-Tcl program, the existing log file will be processed and the report sent. The report generation is part of `sift-tcl`. Usually the configuration specifies the days between reports and the number of days in the report. See the `sift-mail` man page for information on the meta-information syntax. It is basically additional configuration information embedded in Tcl comments that are mostly ignored by the Sift-Tcl interpreter.

SIFT-TCL

The Tcl extensions are patterned after Safe-Tcl. Most of the functions defined in Safe-Tcl are included here, though the syntax is slightly different. Some additional functions for operating on mail folders are also included. Words in ***italics*** are arguments. Items in question marks are optional, though in some cases a minimum of one of the optional items is required.

Several functions require that a message in be specified. Either the whole text of the message can be given as an argument, or a folder handle and a message number can be given. The folder handles are strings returned from the open function. In addition a single message may be read from the standard input by specifying the folder handled as "stdin" and the message number as "1".

There are a few options in the following functions to work with MIME messages. In general MIME is not well supported yet, though it should not be difficult to add. Most of the syntax for options and arguments for MIME is either defined below and doesn't function or can be taken from Safe-Tcl.

When interpreting the Sift-Tcl code, the Tcl variable *inbox* is set to the fully qualified name of the users inbox.

SiftTcl_getheader *field* ?-body *body*? ?-folder *handle* -number *number*?
Returns the requested header field from the specified message or an empty string if the field does not exist. The string returned will include all continuation lines of the header (those lines following the header that begin with blank space).

SiftTcl_getheaders ?-body *body*? ?-folder *handle -number number*?
Returns a list with an element for each header in the specified mail message. Each header is returned as a list with two elements, the header field name and the contents of the header field. If the same header occurs several times it will be in the list several times.

SiftTcl_getaddrs *string*
Parses *string* for e-mail addresses and returns the result as a list. The result is a list even if only one address is returned.

SiftTcl_getaddrprop ***address property***

This returns the request property from the given address. If an empty address is given, the properties returned are for the user's address. The properties follow:

- **proper** Official 822 rendering: *Full Name <user@do.ma.in>*
- **friendly** Returns the full name or the address if there is no full name
- **address** The user name and host: *user@do.ma.in*
- **domain** The domain part of the address
- **mymbox** Returns "1" if address is the recipients, otherwise "0"
- **phrase** The full name or phrase part

SiftTcl_getbodyprop ***part_num property*** **?-body** ***body*****? ?-folder** ***folder*** **-number** ***number*****?**

Returns a specified part of a specified message. *Part_num* must be "1" for the current version. The properties are as follows:

- **all** The full unparsed message header and text
- **headers** The unparsed message headers
- **size** The size of the message (this is only approximate)
- **value** The unparsed body of the message

SiftTcl_open ***folder***

Open the named folder and return a handle. Returns an error if the folder doesn't exist. Use **SiftTcl_mail_append** to create a new folder. See the section below on mail folder naming.

SiftTcl_close ***handle***

Closes the folder open on *handle*. This saves any message status changes and unlocks the folder if the folder format supports locking. All folders are automatically closed when `sift-tcl` exits.

SiftTcl_count ***handle***

Returns the number of messages in the folder open on *handle*.

SiftTcl_getflags ?-body ***body*****? ?-folder** ***folder*** **-number** ***number*****?**

Returns a list of the status flags that are set for the specified message. The possible flags are "DELETED", "ANSWERED", "SEEN", and "FLAGGED". The "SEEN" flag is automatically set when the body of the message is fetched. The others have to be set explicitly. Different mail file storage formats may support different flags and support them in different ways.

SiftTcl_setflag ***flag*** **?-body** ***body*****? ?-folder** ***folder*** **-number** ***number*****?**

Set the given flag for the given mail message.

SiftTcl_clearflag ***flag*** **?-body** ***body*****? ?-folder** ***folder*** **-number** ***number*****?**

Clear the given flag for the given mail message.

SiftTcl_mail_append ***folder*** **?-body** ***body*****? ?-folder** ***folder*** **-number** ***number*****?**

Appends the specified mail message to the given folder. The folder is created if it doesn't exist. See the section below on how the mail folders are named.

SiftTcl_makebody ***content-type ?-id string? ?-parameter string? ?-description string? value ?encoding?***

This constructs a single MIME entity or body. The entity can be sent with **SiftTcl_sendmessage** or used as a component of a multipart message constructed by calling **SiftTcl_makebody** again. If the *content-type* argument is empty then "text/plain" is used. The `-id` argument specifies the Content-ID header and will be generated automatically if not specified. The `-description` argument can be used to specify the Content-Description field. A MIME parameter string for parameters specific to the type of body/entity being created can be given with the `-parameter` argument. Last, the

content encoding can be specified (e.g. “base64”). The actual encoding of the body is not done here and must be done before hand. (**SiftTcl_encode** will be included in a future version for this.)

SiftTcl_makebody ***multipart-content-type*** **?-id** ***string*****?** ***?*****-parameter** ***string*****? ?-description** ***string*****?** ***body ...***
This format is used to construct a multipart MIME entity out of a set of other MIME entities (which may be multipart entities too). The multipart-content-type must begin “multipart/”

SiftTcl_sendmessage -to ***addrlist*** **-subject** ***string*** **-body** ***body*** **?-cc** ***addrlist*****? ?-auxheader** ***name value*****?**
This sends a message. The three arguments `-to`, `-body` and `-subject` are required. The `-to` and `-cc` fields are lists of addresses to which the message is to be sent. The `-body` option gives the body of the message and includes the text, attachments and other sub-parts. It is usually generated with **SiftTcl_makebody**. The `-auxheader` allows any additional headers to be created. These might include “Reply-to:”, “Resent-to:” and “X-loop:”.

SiftTcl_Year
Returns the current year, e.g., 1995.

SiftTcl_Month
Returns the current month with January being 1.

SiftTcl_Day
Returns the current day of the month.

SiftTcl_logmisc ***string***
Places an entry in the log for the most recently retrieved message.

SiftTcl_logerror ***string*******
Places an error entry in the log for the most recently retrieved message.

SiftTcl_logdisposition ***disp***
Sets the end log disposition for a the message most recently retrieved. The disposition is not used by Sift-Tcl itself, but it is used by Sift-Tcl programs and the report generator. Currently it can be unset or set to “FiledInInbox” in order to distinguish messages that were filed in the user’s inbox for the sifting report.

MAIL FOLDER NAMES

Both the open and append functions take a folder name and determine the corresponding file name, mail folder format, and location on the network from it. These names function almost identically as the names in the Pine mail program or the University of Washington IMAP server and fall into several categories:

Local file system
Any name beginning with “/” or “~” is considered to be the name of a file or directory in the local file system that contains stored e-mail. If the folder already exists and the folder format can be determined from file or directory, then that format is used. If the format cannot be determined (the file doesn’t exist, or is zero length) the default driver is used. Currently the MH format and the Berkeley format (common to the “mail” program, Elm, Pine and a few others) are supported and the Berkeley format is the default. See below for the syntax to name an MH folder for creation.

Explicitly selected driver formats
If the name begins with “#”, the part following the “#” selects the driver. This can currently be used to select the MH and Carmel format drivers. The MH syntax is “#mh/*folder*” to access “*folder*”. The MH driver reads the user’s MH profile to determine the path to the MH folders. As mentioned above MH folders can also be opened by naming the particular MH directory in the file system, but MH folders can only be created using the explicit syntax.

The Carmel syntax is "#carmel#*folder*" or "#carmel#*user*#*folder*". The Carmel format stores mail under the `.vmail` directory and is a compatibility driver for a proprietary mail file format.

IMAP access

Folders on IMAP servers can also be accessed with the syntax {*host*}*path*. *Path* is the domain name or IP address of the server, and *path* is the path name on the server. The syntax of the path depends on the server. With the University of Washington IMAP server the entire syntax described in this section can be used in most cases. (e.g., a full file path name, a USENET news group, or a non-fully qualified name.) Two IMAP servers may even be specified with "{*proxy host*}{*host*}*path*" to access one server via another one acting as a proxy.

IMAP servers usually require authentication with a user name and password. This is not supported by `sift-tcl` so the authentication must be done with rimap. For this to work, rimap must be enabled on the server and the `.rhosts` file on the server must allow rsh access for the user.

USENET News

USENET news may be accessed from the local disk, via NNTP or via IMAP. The syntax always begin with "*". To access a news group on the local machine simply prefix it with "*", e.g., "*comp.mail.misc". To access a news group via IMAP on the news server machine use "*{*host*}*news.group*". To use IMAP you must be able to authenticate to the server your `.newsrc` will be stored there. For NNTP the syntax is "*{*host*/nntp}*news.group*", and the `.newsrc` will be stored on the local machine so no authentication is necessary.

Non-fully-qualfied names

If a folder name begins with an alphabetic or numeric character, it is not fully qualified and the driver and format is determined by an automatic process. This depends on the order which the drivers are linked in, whether or not the name matches a file in the current directory and whether the name is valid for a particular driver. It is not recommend that these be used.

Other Drivers

A well written and tested driver for the Tenex mail format is available, but is left out of this distribution because there are name conflicts between it and the Berkeley format driver for non-existent and zero length folders and no explicitly syntax is available to select it. There is also an experimental POP driver available. With a moderate amount of effort other drivers can be created.

SIFT-TCL LIBRARIES

Several Sift-Tcl related libraries are loaded with the Tcl extensions and are generally available for use. Currently, there are two libraries, one used by the code generated by `sift-mail`, and another to maintain a small address database, which is also called by `sift-mail`.

day_of_year ***year month day***

Computes the day number in the year from the year (needed to check for leap year), month and day. The arguments "1995 1 1" will cause 1 to be returned, and "1995 12 31" will cause 365 to be returned.

compare_dates ***date1 date2***

Returns a number less than 0, 0, or greater than 0 if *date1* is less than, equal to, or greater than *date2*. The dates are both lists in the form *{year month day}*.

check_activation ***act_d deact_d***

Returns 1 if the current date is between *act_d* and *deact_d*, and 0 otherwise. The dates are lists in the form *{year month day}*.

check_4_daemon ***addresses***

Check a set of addresses against a known set of addresses for mailer daemons. If any of the addresses match 1, is returned. Otherwise 0 is returned. The check is only a heuristic as there is no

standard address for a mailer daemon. The addresses are matched by a complex regular expression which will match “root”, “postmaster”, “MAILER-DAEMON” and similar strings in the mailbox part of the address, or the full name part.

check_4_list ***folder_handle message_number***
Checks the message specified by the *message_handle* (e.g., as returned by SiftTcl_open) and *message_number* to see if it is from a mailing list. This is by heuristic as the check for the mailer daemon is, since there is no standard for tagging such messages. It checks the “Precedence:” header for “junk”, “bulk”, or “list”. It also checks the address headers for the strings “Multiple Recipients” and “Discussion List.” Capitalization is ignored in all comparisons. 1 is returned if a match is found, and 0 otherwise.

wrap_text ***text***
Given a string (e.g. text of an outgoing message), return a string with all lines wrapped at the first space after 72 characters. The wrapped string is returned. This is used to wrap the lines of outgoing messages. (An alternative is to use MIME quoted-printable or base64 encoding to preserve long lines).

generate_reply_addr ***folder_handle message_numer property***
Return the appropriate address for a reply to the given message. It considers the “Resent-reply-to:”, “Resent-from:”, “Reply-to:” and “From:” headers in that order. The returned address is in the format specified by *property* which is same set of properties as can be specified in the last argument to **SiftTcl_getaddrprop.**

The address database is used to keep track of which addresses an out-of-town notification message has been sent to. Multiple databases can be created and maintained. The records in the databases are keyed by the address and can include additional data for each record. A likely use for the data may be the date the last message was sent to the particular address.

SiftTcl_addrdb_newid
This returns an id for a new database.

SiftTcb_addrdb_delete ***db_id***
Deletes the whole database given by *db_id.*

SiftTcl_addrdb_save ***db_id address ?data?***
Write an entry into the given database. *Address* must not have any spaces in it. (e.g., as returned by SiftTcl_getaddrprop for the “address” property). The data may not have new lines in it.

SiftTcl_addrdb_lookup ***db_id address***
Returns an empty string if *address* is not in the database. Otherwise returns the concatenation of the address and the data fields.

CAVEATS

Several obvious functions are missing. There is no function to parse or generate RFC-822 style dates. Nor is there any function to expunge messages marked as deleted. A number of features to support MIME are missing, such as functions to retrieve the list of body parts, and to encode and decode messages. There are no functions for working on a collection of folders, such one to list a set of folder, or one to delete a folder. These features are available in the *c-client* library upon which most of `sift-tcl` is built. Future work is also needed to support authentication with a user name and password.

Some of the logging functions do not include the message id in the log entries when they could be included.

FILES

$HOME/.forward	Specifies actions for local mail delivery agent

$HOME/.sift-tcl	Default Sift-Tcl code to execute for sifting
$HOME/.siftlog	Logs all mail processed by `sift-tcl`

SEE ALSO

sendmail(8), **smail**(8), **sift-mail**(1)

AUTHOR