Supercite User's Manual

First Edition, Supercite Version 2.3

May 1991

Barry A. Warsaw
bwarsaw@cen.com
...!uunet!cen.com!bwarsaw

Introduction

Supercite is a GNU Emacs package written entirely in elisp which interfaces to common mail and news reading subsystems, and provides sophisticated citing and attributing of the original messages. Supercite has a very specific and limited role in the process of composing replies to both Netnews and Electronic Mail.

Supercite is only useful in conjunction with mail/news reading subsystems such as VM, GNUS, RMAIL, etc. (hereafter referred to collectively as *readers*). Supercite is typically called through a hook, defined by the reader, when the initial reply buffer is set up. Thereafter, supercite's many commands and formatting styles are available in that reply buffer until the reply is sent, at which time supercite is re-initialized and ready for the next reply.

The current version of supercite is 2.3 and this version is known to work with GNU Emacs 18.57 based readers such as RMAIL, RNEWS, and MH-E, as well as all versions of GNUS up to this writing (3.12 and 3.13), GNEWS, all versions of VM up to this writing (VM version 4 from 4.37 to 4.41 and VM version 5) and PCMAIL. Only VM and MH-E (version 3.7 which was released with emacs 18.57) will work with supercite "out of the box." All other readers must overload interfacing routines, supplied with the supercite package, to provide the necessary glue between the reader and supercite. See Chapter 3 [Getting Connected], page 11, for more details.

As of this writing (30-Dec-1992) version 2.3 is now compatible with Lucid Emacs 19.3. This is the only substantive difference between version 2.3 and version 2.2.

Standard operating procedure is usually as follows. You want to reply or followup to an article or message in the reader you are using. Typically, you will enter `r` or `f` to begin composing the reply. The reader you are using will create a buffer and initialize the mail headers appropriately. At this point you will probably be faced with an empty reply buffer. Now you decide that you would like to include part of the original message in your reply so you "yank" the original message into the reply buffer, typically with a key stroke such as `C-c C-y`. This should invoke a reader specific function which fills the buffer with the raw original message and then runs a hook to massage the reply buffer. When you've connected supercite to this hook, it will run at this time, extracting useful information from the various yanked mail headers, creating an attribution string, citing the original message, and inserting a reference header.

Because of this clear division of labor between supercite and the reader, some useful operations, which at first thought should be under the domain of supercite, are really the responsibility of the reader package. For example, many people have indicated that they would like to be able to yank (and cite) only a portion of the original message. Since supercite only modifies the raw text it finds in the buffer as set up by the reader, it is the reader's responsibility to do partial yanking. However, supercite can be told to not modify the text when called via the hook, in which case the raw reply will remain unchanged, but supercite will be initialized for manual citing. See Section 4.1 [Reply Buffer Initialization], page 14.

Another potentially useful thing would be for supercite to set up the outgoing mail headers with information it gleans from the reply buffer composition. But by previously agreed upon convention, any text above the `mail-header-separator` (typically '`--text follows this line--`') which separates mail headers from message bodies is not modifiable by su-

percite. Supercite has no understanding of the semantics of these headers. See Chapter 7 [Hints to Reader Authors], page 20, for more details.

Though supercite is usually loaded and run initially through a hook function, it will "leach" onto the major-mode of the reply buffer by extending the keymap of the buffer, and modifying its major-mode documentation string. In this way, after the initial yank of included text, all of supercite's post-yank formatting commands will be available in the reply buffer, without any advanced knowledge by the reader of supercite's existence or use by you.

Supercite provides routines to do automatic filling of cited text, commands to recite or uncite regions of text in the reply buffer, and commands to perform other beautifications on the reply, maintaining consistent and informative citations throughout. Supercite tries to be as configurable as possible to allow for a wide range of personalized citation styles, but it is also immediately useful with the default configuration, once it has been properly connected to your reader. See Chapter 3 [Getting Connected], page 11, for more details.

1 Citation Styles

A *citation* is the acknowledgment of the original author of a mail message, in the body of the reply. There are two basic citation styles which supercite supports. The first, called *nested citation* style is an anonymous form of citation; in other words, an indication is made that the cited line was written by someone *other* that the current message author (i.e., other than you, composing this reply), but no reference is made as to the identity of the original author. Here's an example of what a message buffer would look like using nested citations after multiple replies:

```
>> John originally wrote this
>> and this as well
> Jane said that John didn't know
> what he was talking about
And that's what I think too.
```

Note that multiple inclusions of the original messages result in a nesting of the "`>`" characters. This can sometimes be quite confusing when many levels of citations are included since it may be difficult or impossible to figure out who actually wrote the first included message. Also, the multiple nesting of "`>`" characters can sometimes make the message very difficult for the eye to scan.

In *non-nested citation* style, each cited line begins with an informative string attributing that line to the original author. Only the first level of attribution will be shown; subsequent citations don't nest the citation strings. The above dialog might look like this when non-nested citation style is used:

```
John> John originally wrote this
John> and this as well
Jane> Jane said that John didn't know
Jane> what he was talking about
And that's what I think too.
```

Notice here that my inclusion of Jane's inclusion of John's original message did not result in a line cited with '`Jane>John>`'.

The supercite variable `sc-nested-citation-p` controls the style of citations. When `nil` (the default), non-nested citations are used. When non-`nil`, nested citations are used.

1.1 Citation Elements

Citation strings are composed of one or more elements. Nested citations, being less complex (and less informative) than non-nested citations, are composed of only a single element, the *citation delimiter*. This string is user defined in the variable `sc-citation-delimiter` and has a default value of `">"`. Nested citations will contain a single space between the last level (oldest) citation and the original text. It is usually a good idea to make `sc-citation-delimiter` a single character.

Non-nested citations are composed of four elements, three of which are directly user definable. The elements are concatenated together, in this order:

1. the *citation leader*. The citation leader is contained in the variable `sc-citation-leader`, and has the default value of a string containing a single tab character.

2. the *attribution string*. This element is supplied automatically by supercite, based on your preferences and the original message's mail headers, though you may be asked to confirm supercite's choice. See Section 1.2 [Attributions], page 4, for more details.
3. the citation delimiter as described above.
4. the *citation separator*. The citation separator is contained in the variable `sc-citation-separator`, and has the default value of a string containing a single space character.

For example, suppose, you were using the default values for the above variables, and supercite provided the attribution string '`Jane`'. In this case, the composed, non-nested citation string used might be something like '`\tJane>` ' (the '`\t`' represents a tab character). This citation string will be inserted in front of every line in the original message that is not already cited. Supercite uses the regular expression in the variable `sc-cite-regexp` to determine whether a line is already cited or not. The default value for this variable is:

```
"\\s *[-a-zA-Z0-9_.]*>+\\s *"
```

Nemacs users may want to set `sc-cite-regexp` to:

```
"\\s *\\([a-zA-Z0-9]\\|\\cc\\|\\cC\\|\\ch\\|\\cH\\|\\ck\\|\\cK\\)*\\s *>+"
```

1.2 Attributions

The attribution string is the part of the author's name that will be used in a composed non-nested citation string to acknowledge the originator of a section of text. Supercite scans the various mail headers present in the original article and uses a number of heuristics to extract strings which it puts into the *attribution list*. This list contains such information as the author's first, middle, and last names, the author's initials, and the author's email terminus. It is consulted during automatic composition of the citation string, and will be the list presented to you if attribution confirmation is selected (also see Chapter 4 [Replying and Yanking], page 14). The variable `sc-preferred-attribution` allows you to determine which part of the author's name should be used when supercite composes automatic citation strings. The value of this variable must be one of the following quoted symbols:

`emailname`
: the author's email terminus.

`initials`
: the author's initials.

`firstname`
: the author's first name.

`lastname`
: the author's last name.

`middlename1`
: the author's first middle name.

Middle name indexes can be any positive integer greater than zero, though it is unlikely that many authors will supply more than one middle name, if that many. Default value for `sc-preferred-attribution` is `'firstname`.

When, for whatever reason, the author's name cannot be found in the '`From:`' mail header, a default author name and attribution string must be supplied. The default author name is set in the variable `sc-default-author-name` (default value is `"Anonymous"`), and

the default attribution string is contained in the variable `sc-default-attribution`, (default value is `"Anon"`). Note that in most circumstances, getting the default author name or attribution is a sign that something is set up incorrectly.

Also, if your preferred attribution cannot be found, or is either `nil` or the empty string, a secondary method can be employed to find a valid attribution string. The variable `sc-use-only-preference-p` controls what happens in this case. If `sc-use-only-preference-p` is non-`nil`, then `sc-default-author-name` and `sc-default-attribution` are used, otherwise, the following steps are taken to find a valid attribution string. The first step to return a non-`nil`, non-empty string becomes the attribution:

1. Use the author's first name.
2. Find the first non-`nil`, non-empty attribution string in the attribution list.
3. If the variable `sc-confirm-always-p` is non-`nil`, then you are queried for an attribution string with a completing read. The possible values for completion are those strings in the attribution list, however, you can override all presented strings by simply typing in your attribution at the prompt.
4. `sc-default-attribution` is used.

Finally, once a legal attribution string is found, you can force the string to lower case characters by setting the variable `sc-downcase-p` to non-`nil`.

1.3 Author Names and Nicknames

Supercite employs a number of heuristics to decipher the author's name base on the '`From:`' field. Supercite can recognize '`From:`' fields with the following forms:

1. `From: John Xavier Doe <doe@speedy.computer.com>`
2. `From: "John Xavier Doe" <doe@speedy.computer.com>`
3. `From: doe@speedy.computer.com (John Xavier Doe)`
4. `From: computer!speedy!doe (John Xavier Doe)`
5. `From: doe%speedy@computer.com (John Xavier Doe)`
6. `From: computer!speedy!doe (John Xavier-Doe)`
7. `From: computer!speedy!doe (John Xavier-Doe -- Decent Hacker)`
8. `From: doe@speedy.computer.com`
9. `From: computer!speedy!doe`

Note that some author fields (as in example 7 above) will contain a descriptive title. The user can choose to ignore the title, while still recognizing hyphenated names (as in the second to last example above), through the use of a regular expression in the variable `sc-titlecue-regexp`. This variable has the default value of `"\\\\s +-+\\\\s +"`.

Some author names may contain non-alphanumeric characters, especially if the author's name is extracted from their email terminus. You can specify that those characters in the author's name be converted to spaces with the variable `sc-spacify-name-chars`. This is a list of characters suitable for `memq`. Default value for this variable is `'(?_ ?* ?+ ?=)`.

For '`From:`' lines matching one of these styles, supercite will be able to pick out the author's full name (i.e., `"John Xavier Doe"`) and the author's email terminus or email name

(i.e., `"doe"`). In examples 8 and 9 above, the author's name will be the email terminus (i.e., `"doe"`). The name extracted by supercite is then split into its individual components and kept in the attribution list either for supercite's automatic use, or for presentation to you in a completing confirmation. When asked to confirm the selected attribution, supercite will present you with this list, but you can add new attributions to the list by just typing it in at the prompt. The selected attribution (whether completed or added) is remembered as the selected attribution for future operations or completions.

Many names also have common nicknames which supercite cannot automatically decipher. Therefore, supercite consults a variable `sc-nicknames-alist` which contains a list of common name to nickname associations. While supercite cannot automatically use a nickname as the attribution string, matching nicknames will be presented to you when attribution confirmation is requested (See Chapter 5 [Post-yank Formatting Commands], page 16). Any part of the author's name can match an entry in this list, and one name can match more than one nickname. The list contains associations of the form:

```
(NAME NICKNAME)
```

Default value for this variable is:

```
'(("Michael" "Mike")
  ("Daniel" "Dan")
  ("David" "Dave")
  ("Jonathan" "John")
  ("William" "Bill")
  ("Elizabeth" "Beth")
  ("Elizabeth" "Betsy")
  ("Kathleen" "Kathy")
  ("Smith" "Smitty"))
```

2 References and Information

Supercite will insert an informative *reference header* at the beginning of the cited body of text, which provides more detail about the original article. Whereas the citation string usually only contains a portion of the original author's name, the reference header can contain such information as the author's full name, email address, the original article's subject, etc. In fact, just about any information contained in the mail headers of the original article can be inserted into a reference header.

There are a number of built-in *header rewrite functions* supplied by supercite, but supercite is extensible in that you can write your own custom header rewrite functions (perhaps using the built-in ones as examples) and tell supercite to use your function. In fact, supercite consults a list of header rewrite functions in the variable `sc-rewrite-header-list`. You can put any rewrite function, custom or built-in, into this list. This list has the default value:

```
'((sc-no-header)
  (sc-header-on-said)
  (sc-header-inarticle-writes)
  (sc-header-regarding-adds)
  (sc-header-attributed-writes)
  (sc-header-verbose)
  (sc-no-blank-line-or-header)).
```

When supercite is called via its hook function `sc-cite-original`, it will automatically call one of these functions to insert a reference header. The one it uses is user definable in the variable `sc-preferred-header-style`. The value of this variable is an integer which is an index into the `sc-header-rewrite-list`, with the first function indexed at zero. The default value for this variable is 1 (i.e., default values use the function `sc-header-on-said` when automatically rewriting the header).

2.1 The Built-in Header Rewrite Functions

Below are examples of the styles of the various built-in header rewrite functions. Please note the following: first, the text which appears in the examples below as '*`fieldkey`*' indicates that the value of a particular *information key* corresponding to '*`fieldkey`*' will be inserted there. (See Section 2.2 [Mail Fields and Information Keys], page 8). For example, in `sc-header-on-said` below, '*`date`*' and '*`from`*' correspond to the values of the '`Date:`' and '`From:`' mail headers respectively.

Also, the string "`>>>>>`" below is really the value of the variable `sc-reference-tag-string`, which is user definable. Finally, the references headers actually written may omit certain parts of the header if the information key associated with *fieldkey* is not present. In fact, for all built-in headers, if the '`From:`' field is not present in the mail headers, the entire reference header will be omitted.

`sc-no-header`

> This function produces no header. It should be used instead of `nil` to produce a blank header. This header can possibly contain a blank line after the `mail-header-separator` line.

`sc-no-blank-line-or-header`
: This function is similar to `sc-no-header` except that any blank line after the `mail-header-separator` line will be removed.

`sc-header-on-said`
```
>>>>> On date, from said:
```

`sc-header-inarticle-writes`
```
>>>>> In article message-id, from writes:
```

`sc-header-regarding-adds`
```
>>>>> Regarding subject; from adds:
```

`sc-header-attributed-writes`
```
>>>>> "sc-attribution" == sc-author <sc-reply-address> writes:
```

`sc-header-verbose`
```
>>>>> On date,
>>>>> sc-author
>>>>> from the organization of organization
>>>>> who can be reached at: sc-reply-address
>>>>> (whose comments are cited below with: "sc-cite")
>>>>> had this to say in article message-id
>>>>> in newsgroups newsgroups
>>>>> concerning the subject of subject
>>>>> see references for more details
```

2.2 Mail Fields and Information Keys

Information keys are nuggets of information that supercite extracts from various mail fields placed in the reply buffer by the reader. Information is kept in a list as *key-value* pairs and can be retrieved for use in reference headers with the function `sc-field`. In addition, other bits of data, composed and created by supercite, are also kept as key-value pairs. In the case of mail fields, the key is always the name of the field, cast to lower case characters, without the trailing colon. Thus, if the following fields were present in the original article:

```
Date: 08 April 1991, 17:32:09 EST
Subject: Better get out your asbestos suit
```

then, `(sc-field "date")` would return the string `"08 April 1991, 17:32:09 EST"`, and `(sc-field "subject")` would return the string `"Better get out your asbestos suit"`. Since the argument to `sc-field` can be any string, it is possible that the mail field will not be present, or that the string was incorrectly typed. In this case, `sc-field` will return a *mumble string* as defined in the variable `sc-mumble-string`. The default value for this variable is the empty string (i.e., `""`).

The variable `sc-mail-fields-list` contains a list of mail fields (as information keys), which supercite will extract for use with `sc-fields`. Only the values of these mail fields will be extracted. Default value for this variable is:

```
'("date"  "message-id"  "subject" "newsgroups" "references"
  "from"  "return-path" "path"    "reply-to"   "organization"
  "reply" )
```

Note that mail headers can also be removed from the body of the reply once their information has been extracted. See Section 4.1 [Reply Buffer Initialization], page 14, for more details.

The '`From:`' field will always be put into the information list exactly once. In addition to these information keys, supercite also always places the following keys into the information list:

`sc-attribution`
: the selected attribution string.

`sc-nested-citation`
: the nested citation string.

`sc-citation`
: the non-nested citation string.

`sc-from-address`
: email address extracted from the '`From:`' field.

`sc-reply-address`
: email address extracted from the '`Reply-To:`' field.

`sc-emailname`
: email terminus extracted from the '`From:`' field.

`sc-initials`
: the author's initials.

`sc-author`
: the author's full name.

`sc-firstname`
: the author's first name.

`sc-lastname`
: the author's last name.

`sc-middlename-1`
: the author's first middle name.

As above, if the author's name has more than one middle name, they will appear as information keys with the appropriate index.

2.3 Electric References

By default, supercite just goes ahead and inserts the reference header indexed by `sc-preferred-header-style`. However, you may want to select different reference headers based on the type of reply or forwarding you are doing. You may also want to preview the reference header before deciding whether to insert it into the reply buffer or not. Supercite provides an optional *electric reference* mode which you can drop into to give you this functionality.

Electric reference mode is a quasi-major-mode which you enter whenever supercite inserts a reference header and the variable `sc-electric-references-p` is non-`nil`. Actually you

are placed into a recursive edit inside the electric reference buffer, which is a read-only buffer.

When in electric reference mode, you can scan back and forth through the list of reference headers in `sc-rewrite-header-list`. You can also set a new preferred header style, jump to any header, or jump to the preferred header. The header will be shown in the electric reference buffer and the header index will appear in the echo area. You cannot, however, actually edit the headers while in electric reference mode; you will have to do that once the header has been inserted into the reply buffer.

The following commands are available while in electric reference mode (shown here with their default key bindings):

`sc-eref-next` (*n*)
: Displays the next reference header in the other buffer. If the variable `sc-electric-circular-p` is non-`nil`, invoking `sc-eref-next` while viewing the last reference header in the list, will wrap around to the first header.

`sc-eref-prev` (*p*)
: Displays the previous reference header in the other buffer. If the variable `sc-electric-circular-p` is non-`nil`, invoking `sc-eref-prev` will wrap around to the last header.

`sc-eref-goto` (*g*)
: Display the reference header indexed by *refnum* where *refnum* is a valid index into `sc-rewrite-header-list`. *refnum* can be supplied as a numeric argument to this command, or you will be queried for it in the minibuffer.

`sc-eref-jump` (*j*)
: Display the preferred reference header – the one indexed by the current value of `sc-preferred-header-style`.

`sc-eref-exit` (*q*, `LFD`, and `RET`)
: Exit from electric reference mode and insert the current header into the reply buffer. Also note that exiting the recursive edit (with command `(exit-recursive-edit)`, typically bound to *ESC C-c*), executes sc-eref-exit.

`sc-eref-abort` (*x*)
: Exit from electric reference mode without inserting the current header.

`sc-eref-setn` (*s*)
: Set the preferred reference header (i.e., `sc-preferred-header-style`) to the currently displayed header.

Supercite will execute the hook `sc-electric-mode-hook` before entering electric reference mode.

3 Getting Connected

Early in supercite's development, the supercite author, many of the major reader subsystem authors, and some supercite users got together and agreed upon a standard interface between all readers and any supercite-like package (of which supercite is the only known one :-). See Chapter 7 [Hints to Reader Authors], page 20, for more information on the details of this interface. You may want to also see Chapter 8 [Thanks and History], page 22, for details on how this interface was proposed and adopted.

Suffice to say that at the time of this writing (May 1991), only VM (all versions after 4.37) and MH-E (version 3.7, which is distributed with emacs 18.57) conform to this interface "out of the box." If you are connecting supercite to one of these two packages, you do not need overloading. All other reader packages must be modified to provide this interface. The file `sc-oloads.el` contains *overloading* routines which will allow you to connect supercite with any of the known major readers currently in existence (see Section 3.2 [Overloading for Selected Readers], page 11). This includes: GNUS (versions 3.12 and 3.13), GNEWS, RMAIL and RNEWS (as distributed with emacs versions after and including 18.55), and PCMAIL. If you are using a reader package other than one of those mentioned above, contact the supercite mailing list about getting or writing overloading functions for that package (see Chapter 9 [The Supercite Mailing List], page 23).

3.1 Quick Guide for All Readers

For any reader package except MH-E, you will need to connect supercite to the standard hook variable `mail-yank-hooks`. MH-E users will need to connect supercite to `mh-yank-hooks`. The supercite function `sc-cite-original` is intended to be run from a hook. It not only cites the text, also does much pre- and post-processing on the reply buffer (see Chapter 4 [Replying and Yanking], page 14) so it should be the first supercite function to be called on a raw reply buffer. Thus, you will need one of the following two lines in your `.emacs` file:

```
(setq mail-yank-hooks 'sc-cite-original) ; for all but MH-E
(setq mh-yank-hooks   'sc-cite-original) ; for MH-E only
```

Also, if supercite is not compiled into your emacs image, you will need to set up the following autoload, also in your `.emacs` file:

```
(autoload 'sc-cite-original "sc" "Supercite 2.3" t)
```

Supercite will run the user definable hook `sc-load-hook` after loading. Default value for this variable is `nil`.

3.2 Overloading for Selected Readers

Most readers contain hardwired citing styles (often unpopular!) which cannot be changed. For these readers, you are required to overload the necessary functions so that they call a hook (i.e., `mail-yank-hooks`) after inserting the original text into the buffer. This hook can then be set to call `sc-cite-original` at the appropriate time.

Once again, note that if you are connecting supercite to either MH-E version 3.7 or VM versions 4.37 and beyond, you do not need overloading and can skip this section.

3.2.1 Quick Guide to Overloading

Since supercite's overloading routines are usually not loaded into your emacs session until they are needed, you should add the following small function to your `.emacs` file. This will load (via the `require`) the supercite overloading package, and then actually overload the functions for the particular reader your are using. Only those original reader functions already bound will be overloaded, and they will only be overloaded once per session.

```
(defun my-sc-overload-hook ()
  (require 'sc-oloads)     ; be sure this file is on your load-path
  (sc-overload-functions))
```

Next, you will need to find a hook in the reader which will be called at an appropriate time to execute `my-sc-overload-hook`. It can be something of a guessing game to find the right hook, but fortunately that's already been done for all the readers supercite currently knows about. You should put one of these two lines in your `.emacs` file:

```
(setq news-reply-mode-hook 'my-sc-overload-hook) ; for RNEWS,GNEWS,GNUS
(setq mail-setup-hook      'my-sc-overload-hook) ; for RMAIL,PCMAIL,GNUS
```

Also for GNEWS users, you need to put the following line in your `.emacs` file:

```
(setq gnews-ready-hook 'my-sc-overload-hook) ; for GNEWS
```

Note that if you have `sc-oloads.el` compiled into your emacs image, you do not need `my-sc-overload-hook`. Instead just use one of the following code fragments:

```
(setq news-reply-mode-hook 'sc-overload-functions) ; for RNEWS,GNEWS,GNUS
(setq mail-setup-hook      'sc-overload-functions) ; for RMAIL,PCMAIL,GNUS
(setq gnews-ready-hook     'sc-overload-functions) ; for GNEWS (need both)
```

3.2.2 Overloading Details

All the overload functions can be found in the file `sc-oloads.el`. In this file are a number of functions which mimic the default behavior of the yanking functions of the major readers. They are typically defined with the same name as their original counterparts, except that "`sc-`" is prepended to the name. At the appropriate time, the supercite version will be overloaded onto the original, by way of `fset`-ing the function cell of the original symbol to the function in the supercite version. In this way, overloading is completely under the control of the individual users, eliminating the need for a system administrator's intervention. Overloading is a general solution to the wider problem of extending the functionality of distribution emacs elisp code, without requiring the editing or patching of the distribution files, something that is often not possible for individual users.

The function `sc-overload-functions` performs the actual `fset` modification, though it does it in a slightly intelligent manner. It will first check to see if the function symbol is bound, and if not, will skip attempting to overload that symbol. Also, it will check to see if overloading has already been performed on the symbol, and will not try to re-overload the function. It does this by setting the *property* `sc-overloaded` on the symbol after overloading it.

The variable `sc-overload-functions` contains an association list of original functions to hook-ified version functions. Entries take the form:

```
(ORIGINAL OVERLOAD)
```

and include all the known yank/reply functions for the major reader subsystems:

- `mail-yank-original`
- `news-reply-yank-original`
- `reply-yank`
- `group-reply-yank`
- `group-follow-yank`

If you encounter a reader package that requires overloading, but that supercite does not currently know about, you will need to write the reply-yanking function which cites the text, to call a hook to do the citing. You should call that hook `mail-yank-hooks` for consistency and set the default value to the default behavior of the reader. Then add the name of the original function and the overloading function to the `sc-overload-functions` association list. See Chapter 7 [Hints to Reader Authors], page 20, for more details.

4 Replying and Yanking

When you perform a yank in the reply buffer set up by your reader, it should execute the function `sc-cite-original` via one of the hooks mentioned above (see Chapter 3 [Getting Connected], page 11). Since this function is called by a hook, it is not passed any arguments and so expects the raw reply buffer to be very specifically formatted (see Chapter 7 [Hints to Reader Authors], page 20). For the most part, `sc-cite-original` will do some default things, based on your preferences, with very little direct interaction with you. After the initial yank of the original message, you can use various supercite commands to reformat and beautify your reply (see Chapter 5 [Post-yank Formatting Commands], page 16).

4.1 Reply Buffer Initialization

Executing `sc-cite-original` performs the following initializations of the reply buffer:

1. *Runs* `sc-pre-hook`. You can set this variable to execute any function. It will be called before supercite does anything. You could conceivably use this hook to set certain supercite variables based on the reply buffer's mode or name (i.e., to do something different based on whether you are replying or following up to an article).
2. *Gets information from the mail headers.* All previously retrieved information keys are deleted, then the mail headers in the body of the text are scanned. Information key-value pairs are created for each header found. For example, such things as the author's name and email address are extracted, and the attribution and citation strings are also derived at this point. If the variable `sc-confirm-always-p` is set, supercite will confirm the selected attribution string with you at this time before it uses it in the citation string.
3. *"Nukes" the mail headers.* If the variable `sc-nuke-mail-headers-p` is non-`nil`, the mail headers **in the body of the message**, will be deleted. As mentioned before, supercite does nothing to the mail headers above the `mail-header-separator` line. You can control which mail headers are kept and which are deleted by modifying the variable `sc-header-nuke-list`. This variable contains a list of mail headers to remove, where each entry in the list is a self contained regular expression unit. These units will be `concat`'ed together into one big regular expression of the form:

   "^*header*:\\|^*header*:\\|...^*header*:"

   The entries in `sc-header-nuke-list` correspond to the individual *header*'s in the above example, and the entries are case insensitive. If `sc-header-nuke-list` is `nil`, no headers will be removed (however, it is better to use `sc-nuke-mail-headers-p` for this). The default value of `sc-header-nuke-list` is:

```
'("via" "origin" "status" "received" "remailed" "cc" "sender" "replied"
  "organization" "keywords" "distribution" "xref" "references" "expires"
  "approved" "summary" "precedence" "subject" "newsgroup[s]?"
  "\\(followup\\|apparently\\|errors\\|\\(\\(in-\\)?reply\\)?-\\)?to"
  "x-[a-z0-9-]+" "[a-z-]*message-id" "\\(summary-\\)?line[s]"
  "\\(\\(return\\|reply\\)-\\)?path" "\\(posted-\\)?date"
  "\\(mail-\\)?from")
```

4. *Cites the message body.* If the variable `sc-all-but-cite-p` is non-`nil`, the message will not be cited. This way, you can have supercite initialize itself and do everything but cite the text. This would be useful if you were citing a very long article (which may take a bit of time), or for some unique citing formats (i.e., mixed text and code, with different filling and citing requirements).
5. *"Leaches" onto the current buffer.* Because supercite is intended to run with a number of different readers, many of which will have been designed without knowledge of supercite, and because the supercite package adds functionality to these subsystems, it must be somewhat subversive in the way it adds its functions to the keymaps of the current buffers, and its documentation to the documentation strings of the major-mode of the reply buffer. See Chapter 6 [Keymaps], page 18, for more information on how supercite sets the keymap of the reply buffer and how you can change the default key bindings.

   Once supercite has attached itself to the reply buffer, getting major-mode help, by typing `C-h m` (`describe-mode`) will print out not only the original mode documentation string, but also supercite's documentation string. This documentation will describe the available supercite commands and their key bindings.
6. *Runs* `sc-post-hook`. This variable is very similar to `sc-pre-hook`, except that it runs after `sc-cite-original` is finished. This hook is provided mostly for completeness and backward compatibility. Perhaps it could be used to reset certain variables set in `sc-pre-hook`.

4.2 Filling Cited Text

Supercite provides some paragraph filling functions and will automatically fill paragraphs as they are cited, when the variable `sc-auto-fill-region-p` is non-`nil` (the default value). There are other packages freely available which work quite well when filling paragraphs of the non-nested citation style (and probably better than supercite's built-in functions!) so supercite calls the filling function via the hook `sc-fill-paragraph-hook`. The default is for supercite to call the function `sc-fill-paragraph`.

You can also have supercite query you before filling each paragraph in the cited region. This would be useful for citing an article with mixed code and text, where you would want to fill the text regions but not the code regions. Set the variable `sc-auto-fill-query-each-paragraph-p` to non-`nil` (default value is `nil`) for this feature. Note that `sc-auto-fill-region-p` must be non-`nil` for query filling to work.

Note further that turning off auto-filling does not preclude you from manually filling each paragraph (see Chapter 5 [Post-yank Formatting Commands], page 16).

Finally, supercite will collapse leading whitespace between the citation string and the text line when the variable `sc-fixup-whitespace-p` is non-`nil`. The default value for this variable is `nil`.

5 Post-yank Formatting Commands

Once the original message has been yanked into the reply buffer, and `sc-cite-original` has had a chance to operate on the buffer, a number of useful supercite commands will be available to you. These commands are described in this section. Note that the key bindings given with the command names are those for the default keymap in `sc-default-keymap`. See Chapter 6 [Keymaps], page 18, for more information on changing the key bindings.

5.1 Commands to Manually Cite, Recite, and Uncite

Probably the three most common post-yank formatting operations that you will perform will be the manual citing, reciting, and un-citing of regions of text in the reply buffer. Often you may want to recite a paragraph to use a nickname, or manually cite a paragraph when using `sc-all-but-cite-p`. The following commands perform these functions on the region of text between *point* and *mark*. Each of them sets the *undo boundary* before modifying the region so that the command can be undone in the standard emacs way.

`sc-cite` (*C-c C-t*)
: This command cites each line in the region of text, but only if the line is not already cited as described by `sc-cite-regexp`. It also inserts a reference header at the top of the region. If you supply the optional numeric argument, it will be passed to `sc-insert-reference` (see Section 5.2 [String Insertion Commands], page 16). You will always be asked to confirm the attribution string before the region is cited, regardless of the value of `sc-confirm-always-p`.

`sc-uncite` (*C-c C-u*)
: This command removes any citation strings from the beginning of each cited line in the region.

`sc-recite` (*C-c C-a*)
: This command simply un-cites, then cites the lines in the region, asking for confirmation of the new attribution string.

5.2 String Insertion Commands

These two functions insert strings into the reply buffer.

`sc-insert-reference` (*C-c C-r*)
: Inserts a reference header into the reply buffer at point. With no arguments, the header indexed by `sc-preferred-header-style` is inserted. An optional numeric argument is the index into `sc-rewrite-header-list` indicating which reference header to write. Of course, if `sc-electric-references-p` is set, you are dropped into electric reference mode instead.

  With just the universal argument (*C-u*), electric reference mode is entered, regardless of the value of `sc-electric-references-p`.

`sc-insert-citation` (*C-c C-i*)
: Inserts the current citation string at the beginning of the line that point is on.

5.3 Information Commands

These commands allow you to modify and view various bits of information.

`sc-modify-information` (*C-c C-m*)
: Allows you to interactively modify information key value pairs (See Section 2.2 [Mail Fields and Information Keys], page 8). With the universal argument (*C-u*), it deletes a key (and its associated value) instead. This command will ask for completion on the information key to modify. You can add a new key-value pair by supplying a new key string at the prompt instead of completing.

`sc-view-field` (*C-c f*)
: Allows you to simply view information key values. This is essentially an interactive version of `sc-field`. It will prompt you for the information key to view. With the universal argument (*C-u*), this command will also insert the key value into the current buffer at point.

`sc-glom-headers` (*C-c g*)
: Lets you re-initialize supercite's information key-value pairs from a set of mail headers in the region between point and mark. This function is especially useful for replying to digest messages where supercite will initially set up its information for the digest originator, but you want to cite each component article with the real message author. Note that unless an error during processing occurs, any old information is lost.

`sc-version` (*C-c C-v*)
: Shows the version of supercite you are using. With the optional universal argument, this command inserts the version information into the current buffer (good for identifying bug reports! – see Chapter 9 [The Supercite Mailing List], page 23).

5.4 Miscellaneous Commands

`sc-open-line` (*C-c C-o*)
: Similar to emacs' standard `open-line` commands, but inserts the citation string in front of the new line. As with `open-line`, an optional numeric argument inserts that many new lines.

`sc-fill-paragraph-manually` (*C-c q* and *C-c C-q*)
: Manually fills the current paragraph. Actually this is an interactive version running the hook `sc-fill-paragraph-hook`, however it does bind the global variable `sc-fill-arg` to the value of the optional argument. What this implies of course, is that you can use any paragraph filling package you want to actually do the filling of the cited paragraph. If that package takes an argument, you can write a simple wrapper function to pass `sc-fill-arg` as that argument.

`sc-describe` (*C-c ?*)
: This function has been obsoleted by the texinfo manual you are now reading. It is still provided for compatibility, but it will eventually go away.

6 Keymaps

Since every reader can conceivably use a different buffer and/or major-mode to reply in, supercite can't know ahead of time what the state of the buffer is. So supercite needs to leach onto whatever buffer the reply is being made in, modifying the keymap and the documentation string for that buffer. A mechanism was developed to provide a *per-interface* keymap, which installs itself into the reply buffer's `current-local-map` based on the major-mode of the buffer.

The variable `sc-local-keymaps` contains an association list of the form:

```
((MAJOR-MODE [FUNCTION | MAJOR-MODE])+)
```

When it is time to modify the keymap of the current buffer, supercite looks up that buffer's major-mode in this association list. If it matches an entry, supercite looks at the value associated with the key. If the value is a list, it is assumed that this list is a function which will set the current local keymap as intended. If however, the value is another major-mode symbol name, then this returned major mode is looked up in `sc-local-keymaps`, and the resulting keymap-setting function is evaluated. Only one level of indirection is allowed, but this does let you save space when defining key bindings. If you have many modes which have the same bindings, you only need define the keymap setting function once, and then let all other modes refer to this mode's keymap.

Here is the default value for `sc-local-keymaps`:

```
'((mail-mode
   (lambda ()
     (local-set-key "\C-c\C-r" 'sc-insert-reference)
     (local-set-key "\C-c\C-t" 'sc-cite)
     (local-set-key "\C-c\C-a" 'sc-recite)
     (local-set-key "\C-c\C-u" 'sc-uncite)
     (local-set-key "\C-c\C-i" 'sc-insert-citation)
     (local-set-key "\C-c\C-o" 'sc-open-line)
     (local-set-key "\C-c\C-q" 'sc-fill-paragraph-manually)
     (local-set-key "\C-cq"    'sc-fill-paragraph-manually)
     (local-set-key "\C-c\C-m" 'sc-modify-information)
     (local-set-key "\C-cf"    'sc-view-field)
     (local-set-key "\C-cg"    'sc-glom-headers)
     (local-set-key "\C-c\C-v" 'sc-version)
     (local-set-key "\C-c?"    'sc-describe)
   ))
  (mh-letter-mode
   (lambda ()
     (local-set-key "\C-c\C-r" 'sc-insert-reference)
     (local-set-key "\C-c\C-t" 'sc-cite)
     (local-set-key "\C-c\C-a" 'sc-recite)
     (local-set-key "\C-c\C-u" 'sc-uncite)
     (local-set-key "\C-ci"    'sc-insert-citation)
     (local-set-key "\C-c\C-o" 'sc-open-line)
     (local-set-key "\C-cq"    'sc-fill-paragraph-manually)
     (local-set-key "\C-c\C-m" 'sc-modify-information)
```

```
      (local-set-key "\C-cf"    'sc-view-field)
      (local-set-key "\C-cg"    'sc-glom-headers)
      (local-set-key "\C-c\C-v" 'sc-version)
      (local-set-key "\C-c?"    'sc-describe)
    ))
  (news-reply-mode mail-mode)
  (vm-mail-mode mail-mode)
  (e-reply-mode mail-mode)
  (n-reply-mode mail-mode)
  )
```

If the major-mode of the buffer is not found in the association list, then the function in `sc-default-keymap` is evaluated. The default value for `sc-default-keymap` is:

```
'(lambda ()
   (local-set-key "\C-c\C-r" 'sc-insert-reference)
   (local-set-key "\C-c\C-t" 'sc-cite)
   (local-set-key "\C-c\C-a" 'sc-recite)
   (local-set-key "\C-c\C-u" 'sc-uncite)
   (local-set-key "\C-c\C-i" 'sc-insert-citation)
   (local-set-key "\C-c\C-o" 'sc-open-line)
   (local-set-key "\C-c\C-q" 'sc-fill-paragraph-manually)
   (local-set-key "\C-cq"    'sc-fill-paragraph-manually)
   (local-set-key "\C-c\C-m" 'sc-modify-information)
   (local-set-key "\C-cf"    'sc-view-field)
   (local-set-key "\C-cg"    'sc-glom-headers)
   (local-set-key "\C-c\C-v" 'sc-version)
   (local-set-key "\C-c?"    'sc-describe)
   )
```

The keymap for electric reference mode can also be user defined, but since there is no need for a per-interface map, there is only a single variable, `sc-electric-mode-map` which contains the keymap. If you set this variable, you can override the default key bindings for electric reference mode. See Section 2.3 [Electric References], page 9, for a description of the default key bindings for this mode.

7 Hints to Reader Authors

In June of 1989, some discussion was held between the various reader authors, the supercite author, and other supercite users. These discussions centered around the need for a standard interface between these readers and supercite (or any future readers, or supercite-like packages). This interface was formally proposed by Martin Neitzel on Fri, 23 Jun 89, in a mail message to the supercite mailing list:

```
Martin> Each news/mail-reader should provide a form of
Martin> mail-yank-original that

Martin> 1: inserts the original message incl. header into the
Martin>    reply buffer; no indentation/prefixing is done, the header
Martin>    tends to be a "full blown" version rather than to be
Martin>    stripped down.

Martin> 2: ‘point’ is at the start of the header, ‘mark’ at the
Martin>    end of the message body.

Martin> 3: (run-hooks 'mail-yank-hooks)

Martin> [Supercite] should be run as such a hook and merely
Martin> rewrite the message.  This way it isn't anymore
Martin> [supercite]'s job to gather the original from obscure
Martin> sources. [...]
```

This proposal was adopted, and thus supercite 2.3 conforms to this interface, as does VM (versions 4.37 and beyond, including versions 5.xx), as well as MH-E 3.7 which is distributed with emacs 18.57.

If you are writing a new reader package, or updating an existing reader package, you should make it conform to this interface so that your users will be able to link supercite easily and seamlessly. To do this, when setting up a reply or forward buffer, your reader should follow these steps (outlined above and discussed in greater detail below):

1. Insert the original message, including the mail headers into the reply buffer. At this point you should not modify the raw text in any way, and you should place all the original headers into the body of the reply. This means that many of the mail headers will be duplicated, one copy above the `mail-header-separator` line and one copy below, however there will probably be more headers below this line.
2. Set “point” to the beginning of the line containing the first mail header in the body of the reply. Set “mark” to the end of the message text. It is very important that the region be set around the text supercite is to modify and that the mail headers are within this region. Supercite will not look at or modify anything outside the region, and anything within the region is fair game, so don't put anything that **must** remain unchanged inside the region.
3. Run the hook `mail-yank-hooks`. You will probably want to provide some kind of default citation functions in cases where the user does not have supercite installed. By default, your reader should set `mail-yank-hooks` to execute this default citation

function. Users who want to use supercite can then set `mail-yank-hooks` to `sc-cite-original`.

If you do all this you will not need to provide overloading routines and your reader will join the ranks of those subsystems who conform to this interface "out of the box."

8 Thanks and History

The supercite package was derived from its predecessor superyank 1.11 which was inspired by various bits of code and ideas from Martin Neitzel and Ashwin Ram. They were the folks who came up with the idea of non-nested citations and implemented some rough code to provide this style. Superyank and now supercite 2.3, have evolved to the point where much of the attribution selection mechanism is automatic, and features have been continuously added through the comments and suggestions of the supercite mailing list participants.

Many of these folks have contributed their help in debugging, making suggestions for enhancements, and supplying support code or bug fixes for the previous versions of supercite. I would like to thank (in alphabetical order):

- Mark D. Baushke
- Chris Davis
- Dan Jacobson
- Kyle Jones
- David Lawrence
- Piet van Oostrum
- Khalid Sattar
- Kayvan Sylvan
- Masanobu Umeda
- Joe Wells

I apologize if I've left anybody out. All who have helped have been greatly appreciated.

Also, thanks to David Eckelkamp, John Stoffel, and Walter Rowe for proof reading the initial drafts of this manual.

9 The Supercite Mailing List

The author runs a simple mail expanding mailing list for discussion of issues related to supercite. This includes enhancement requests, bug reports, general help questions, etc. To subscribe or unsubscribe to the mailing list, send a request to the administrative address:

```
Internet: supercite-request@anthem.nlm.nih.gov
    UUCP: uunet!anthem.nlm.nih.gov!supercite-request
```

Please be sure to include the most reliable and shortest (preferably internet) address back to you. To post articles to the list, send your message to this address (you do not need to be a member to post, but be sure to indicate this in your article or replies may not be CC'd to you):

```
Internet: supercite@anthem.nlm.nih.gov
    UUCP: uunet!anthem.nlm.nih.gov!supercite
```

If you are sending bug reports, please indicate the version of supercite and emacs that you are using, and the name and version number of the reader package you're trying to interface with. Without this information, it will be difficult to get useful advice.

Concept Index

Command Index

Since all supercite commands are prepended with the string "sc-", each appears under its sc-*command* name and its *command* name.

V

Key Index

Variable Index

Since all supercite variables are prepended with the string "sc-", each appears under its sc-*variable* name and its *variable* name.

S

T

U

Short Contents

Table of Contents