%% %% This is file ‘x.tex’, %% generated with the docstrip utility. %% %% The original source files were: %% %% fileerr.dtx (with options: ‘exit’) %% %% This is a generated file. %% %% Copyright 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 %% 2006 2008 2009 %% The LaTeX3 Project and any individual authors listed elsewhere %% in this file. %% %% This file was generated from file(s) of the Standard LaTeX ‘Tools Bundle’. %% ———————————————————————— %% %% It may be distributed and/or modified under the %% conditions of the LaTeX Project Public License, either version 1.3c %% of this license or (at your option) any later version. %% The latest version of this license is in %% http://www.latex-project.org/lppl.txt %% and version 1.3c or later is part of all distributions of LaTeX %% version 2005/12/01 or later. %% %% This file may only be distributed together with a copy of the LaTeX %% ‘Tools Bundle’. You may however distribute the LaTeX ‘Tools Bundle’ %% without such generated files. %% %% The list of all files belonging to the LaTeX ‘Tools Bundle’ is %% given in the file ‘manifest.txt’. %% \batchmode \errmessage\csname @end\endcsname \end \endinput %% %% End of file ‘x.tex’.

This file documents emacs-w3m, an Emacs interface to w3m.

This edition is for emacs-w3m version [No value for “VERSION”].

Emacs-w3m User’s Manual

An Emacs interface to w3m for emacs-w3m version [No value for “VERSION”]

The emacs-w3m development team

Table of Contents

Emacs-w3m User's Manual

The emacs-w3m development team

This manual corresponds to emacs-w3m version [No value for "VERSION"].

1 Preliminary remarks

Emacs/W3 once was the most popular web browser on Emacs. However, it worked so slowly that we wanted a speedy alternative. On the other hand, w3m was a pager with WWW capability, developed by Akinori ITO. Although it was a pager, it was possible to use it as a text-mode WWW browser, so we started developing an Emacs interface to w3m.

Our special thanks go to Akinori ITO and the w3m team for the excellent w3m program. We would also like to thank everybody who has submitted comments, suggestions, and bug fixes. Even though we're not aware of any problems, all responsibility for this program is ours (the emacs-w3m development team), but there is absolutely no warranty. The emacs-w3m program was first created by TSUCHIYA Masatoshi in June 2000.

See also the emacs-w3m official page.

2 It's so easy to begin to use emacs-w3m

Emacs-w3m may have already been installed on your system, in which case you can skip this section and begin to use the program at once. If you're not that lucky, read on to learn how to install emacs-w3m.

2.1 What version of Emacs can be used?

You can run emacs-w3m in various versions of Emacsen listed below:

'`Emacs 21.1 or greater`'
: No additional packages are required.

'`XEmacs 21.x`'
: First of all, you should note that emacs-w3m supports only XEmacs 21.4.17 and later and XEmacs 21.5-b19 and later. In addition, you need to have installed the latest xemacs-base package including the '`timer-funcs.el`' module.

  The APEL package and the '`gifsicle`' program are required. In addition, it would be better to have installed the '`rfc2368.el`' module which parses '`mailto`' urls (see Section 2.3 [Other Requirements], page 3).

'`Emacs 20.x, Emacs 19.34 (including Mule 2.3)`'
: Emacs-w3m no longer supports those Emacs versions.

If you use the development version of GNU Emacs, perhaps you should run the CVS version of emacs-w3m on it. In that case, it is strongly recommended that you join the Chapter 11 [Mailing List], page 94.

2.2 Using w3m: the reason why emacs-w3m is fast

Emacs-w3m uses the external w3m program as a back-end to retrieve web contents and as an HTML rendering engine; that's how we could create an accelerated Emacs web browser with asynchronous operation.

You must install the latest w3m, it is available at:

`http://prdownloads.sourceforge.net/w3m/`

2.3 Things required to run emacs-w3m

Depending on the Emacs version you're using, third party packages may be required. This section provides resources to help you find and install them.

'`APEL`'
: Indispensable to XEmacs. You should install APEL before building emacs-w3m. APEL is available at:

  `http://kanji.zinbun.kyoto-u.ac.jp/~tomo/lemi/dist/apel/`

  Note that you must not use the APEL XEmacs package (which is contained in SUMO) of the versions older than 1.32. If you have already installed such a version, you should upgrade it or use the following directives to replace it with APEL which is linked above (you can also use the same directives in order to newly install APEL):

```
% rm -fr /usr/local/lib/xemacs/xemacs-packages/lisp/apel
% cd apel-10.7
% make install-package XEMACS=xemacs-21.4.x\
  PACKAGEDIR=/usr/local/lib/xemacs/xemacs-packages
```

'gifsicle'

Indispensable to XEmacs. There is a known bug in all XEmacs 21.x series that won't let it display optimized animated gifs correctly or may make it crash when some kind of an interlaced gif image is displayed. Emacs-w3m uses the '**gifsicle**' program to convert gif data in order to make it possible to be handled by XEmacs 21.x. It is available at:

http://www.lcdf.org/gifsicle/

'ImageMagick'

If the '**convert**' program bundled with the ImageMagick package is available on your system, emacs-w3m will use it to resize images or to convert '**favicon**' images into a format Emacs can handle. Emacs-w3m will work without ImageMagick, but installing it will improve your surfing experience. You can get the ImageMagick package from:

ftp://ftp.imagemagick.org/pub/ImageMagick/

To manipulate '**favicon**' images, we recommend version 5.4.0-5 and later, previous versions may work but we didn't check them thoroughly.

'FLIM'

The FLIM package is required to use the '**shimbun**' library. The '**shimbun**' library is a collection of tools for reading web newspapers, you can use it with Gnus, Mew or Wanderlust. See Chapter 9 [Shimbun Library], page 59.

Note that the FLIM package requires the APEL package regardless of the version of Emacs you are using. Therefore, you must install both APEL and FLIM if you would like to use the '**shimbun**' library. The FLIM package is available at:

http://kanji.zinbun.kyoto-u.ac.jp/~tomo/lemi/dist/flim/flim-1.14/

'Mule-UCS'

If you use XEmacs 21.4, or need to read Chinese text, Japanese text, Korean text, etc. using Emacs 21, we recommend you install the Mule-UCS Emacs Lisp package so as to enable emacs-w3m to display pages encoded by UTF-8, which is the typical coding system of the Unicode system. Note that Emacs 21 supports the Unicode system partially (e.g., for Latin text) and Emacs 22 or greater fully supports it. The Mule-UCS Emacs Lisp package is available at:

http://unit.aist.go.jp/itri/itri-gist/ftp.m17n.org/pub/mule/Mule-UCS/test/Mule-UC

(The official page for Mule-UCS will open in the near future in http://www.meadowy.org/.)

If you use it with Emacs 21.2, it is necessary to apply this patch before installation.

If you are an XEmacs 21.4 user, the compiled package is here:

ftp://ftp.xemacs.org/xemacs/packages/mule-ucs-1.14-pkg.tar.gz

(You don't need to install it in addition to Mule SUMO, that contains it.)

'codepage-ex'
: Some web sites in Europe specify a 'charset=ISO-8859-1' encoding, but really use windows-1252. Since iso-8859-1 is insufficient to decode that, emacs-w3m uses windows-1252 as a superset of iso-8859-1 if it is available. The windows-1252 coding system is built-in since Emacs 22. Even in Emacs 21, you can enable to use it using the module called codepage-ex. See the following page:

  http://nijino.homelinux.net/emacs/codepage-ex.html

'rfc2368.el'
: Install it if you'd like to enable emacs-w3m running with XEmacs to parse 'mailto' urls properly. You can find the 'rfc2368.el' module which can be used with XEmacs in the 'attic' directory in the emacs-w3m distribution. To install it, copy the 'rfc2368.el' file to the directory belonging to the load-path, and perform the following command:

```
% xemacs -batch -vanilla -f batch-byte-compile rfc2368.el
```

2.4 Installing emacs-w3m

See the official page http://emacs-w3m.namazu.org/ for instructions on how to get the latest emacs-w3m. Is everything ready? Now, let's begin.

In order to install emacs-w3m on non-UNIX-like systems (or any system lacking the ability to execute the 'configure' script or has no 'make' command), skip this section and go to the next section See Section 2.5 [Non-UNIX-like systems], page 6.

1. First, extract a tarball of the emacs-w3m distribution and enter the top directory as follows:

   ```
   % tar zxf emacs-w3m-[No value for ``VERSION'']].tar.gz
   % cd emacs-w3m-[No value for ``VERSION'']]
   ```

   If you've checked out emacs-w3m from CVS, you have to run 'autoconf' with no argument to generate the 'configure' script.

2. Run the 'configure' script.

   ```
   % ./configure
   ```

   Important notice to Gnus users:

   If multiple versions of Gnus are installed on your system (it is likely that there are the released version and the development version of Gnus), make sure that the load-path contains the directory where the version you use is installed (check for the 'gnus.elc' file). You can ensure that with the '--with-addpath' option as follows:

   ```
   % ./configure --with-addpath=/usr/local/share/emacs/site-lisp/gnus
   ```

   If you've installed APEL, FLIM or something in

   non-standard directories other than the default load-path, you must specify them using the '--with-addpath' option as follows (you may also include the Gnus directory in it separated with ':'):

   ```
   % ./configure --with-addpath=/opt/share/apel:/opt/share/flim
   ```

3. Just run 'make' and 'make install'. See also the next step if you are using XEmacs.

   ```
   % make
   ```

```
% make install
```

All Lisp and info files will be installed in the appropriate directories. Now, how do you know what files will go where? To know it beforehand, use this:

```
% make what-where
```

If you are using Emacs or XEmacs capable of displaying images, you had better install icon image files. To do this:

```
% make install-icons
```

or

```
% make install-icons30
```

The later is for using the slightly larger icon images.

4. You can also install emacs-w3m as an XEmacs package using '`make install-package`' instead of '`make install`' as follows:

```
% make
% make install-package
```

In this case, you don't have to execute '`make install-icons`' nor '`make install-icons30`'.

If you need to specify the package directory, there are two ways to do that:

- Use the '`configure`' option '`--with-packagedir=`'. For example:

```
% ./configure --with-packagedir=/opt/xemacs/xemacs-packages
% make what-where
% make
% make install-package
```

- Use the '`PACKAGEDIR`' variable. For example:

```
% ./configure
% make what-where PACKAGEDIR=/opt/xemacs/xemacs-packages
% make
% make install-package PACKAGEDIR=/opt/xemacs/xemacs-packages
```

2.5 Installing on non-UNIX-like systems

If you cannot execute the '`configure`' script on your system, or if no '`make`' command is available, cast the following spell:

```
% emacs -batch -q -no-site-file -l w3mhack.el NONE -f w3mhack-nonunix-install
```

If APEL, FLIM (or any other library) aren't installed in the ordinary places, the installer will leave them out. In such a case, it is necessary to tell those places to the installer as shown below:

```
% emacs -batch -q -no-site-file -l w3mhack.el //c/share/apel://c/share/flim -f w3mhack
```

2.6 Minimal settings to run emacs-w3m

This section mentions some fundamental settings for emacs-w3m. If you want to fine-tune your installation, you'll find many customizable variables in Chapter 5 [Customizable Variables], page 30.

'Autoloads'
: You don't need this if you've installed emacs-w3m as an XEmacs package (see Section 2.4 [Installing Emacs-w3m], page 5) because the 'w3m/auto-autoloads.el' takes care of setting up autoloads.

  In all other cases, put the following line in your '~/.emacs' file:

```
(require 'w3m-load)
```

'Startup File'
: We recommend using the '~/.emacs-w3m' file (which is the default value of w3m-init-file) if you need to twiddle some emacs-w3m variables. This file is similar to '~/.emacs', but is read when emacs-w3m starts. Note that some options shouldn't be modified there, for example, w3m-command.

'Proxy Gateway'
: If you are behind a firewall and access the Internet through a proxy gateway, you need to instruct w3m to use it.

  There are several ways to do this, one is to set the http_proxy environment variable globally in the shell something like:

```
setenv http_proxy http://proxy.hogege.com:8000/
```

  Another way is to customize the w3m-command-arguments variable to add the options '-o' and 'http_proxy=http://PROXY_SERVER_NAME:PORT/'.

  This can also be done in your '~/.emacs-w3m' file as shown below:

```
(setq w3m-command-arguments
      (nconc w3m-command-arguments
             '("-o" "http_proxy=http://proxy.hogege.com:8000/")))
```

  To specify hosts for which the proxy shouldn't be used (Intranet sites and the like), set the no_proxy (note that it is not no-proxy) environment variable to a comma-separated list of hostnames. Alternatively, you can set the w3m-no-proxy-domains variable to a list of domain names (not host names) as follows:

```
(setq w3m-no-proxy-domains '("local.com" "neighbor.com"))
```

  See also the documentation of the w3m-command-arguments-alist variable for instructions on how to use regexps to specify no_proxy hosts.

3 Basic usage

3.1 Let's go netsurfing!

You can, by the keys, let emacs-w3m do all the web browsing operations. Emacs-w3m uses the '`Lynx-like`' keymap (see Section 3.1.1 [Key Binding], page 8) by default. Of course, you can use the mouse buttons, too.

3.1.1 There are two types of the key bindings

Since emacs-w3m is a late-coming web browser in the history of Emacs web browsers, we offer two types of the key bindings in order that users can get used to the new web browser easily. One is called the '`Lynx-like`' keymap, the other is the '`Info-like`' keymap. You can see what they mean from those names, can't you? The former is similar to that of '`Lynx`' which is the text-based web browser, and the later is similar to that of '`Info`' which is GNU's official document browser. By default, the '`Lynx-like`' keymap is used. If you would like to use the '`Info-like`' keymap, type

M-x customize-option ⟨RET⟩ *w3m-key-binding* ⟨RET⟩

choose the '`Info-like`' keymap, and save the changed state. Otherwise, add the following snippet to your '`~/.emacs`' file, not '`~/.emacs-w3m.el`' file:

```
(setq w3m-key-binding 'info)
```

To change the key binding one by one, modify the `w3m-mode-map` variable in your '`~/.emacs-w3m.el`' file like the following:

```
(define-key w3m-mode-map [up] 'previous-line)
(define-key w3m-mode-map [down] 'next-line)
(define-key w3m-mode-map [left] 'backward-char)
(define-key w3m-mode-map [right] 'forward-char)
```

3.1.2 Go ahead, just try it

You can invoke emacs-w3m using three different commands, listed below. Try one of these commands: a web page is displayed in an Emacs buffer named "*w3m*", meaning that it is an emacs-w3m buffer. As you will probably notice, the major mode for an emacs-w3m buffer is `w3m-mode`, there are also minor modes (see Section 3.4 [Managing Bookmarks], page 17).

This section explains the most fundamental usage of these commands, see Chapter 12 [Emacs-w3m Functions], page 95 for more information on them. See also Chapter 5 [Customizable Variables], page 30 for variables you can use to customize emacs-w3m's behavior.

`w3m` Start emacs-w3m, displaying the homepage specified in the `w3m-home-page` variable. The default value for `w3m-home-page` is "about:". Set the `w3m-quick-start` variable to `nil` if you want to input a target URL every time you start emacs-w3m.

You can also use this as an argument when starting emacs. Examples:

```
% emacs -f w3m
```

To specify a URL, you could also use:

```
% emacs -f w3m http://emacs-w3m.namazu.org/
```

`w3m-find-file`
: Prompt for a local file name in the minibuffer, and display it in emacs-w3m.

`w3m-browse-url`
: Prompt for a URL in the minibuffer, and display it in emacs-w3m. This command is provided, if anything, in order to start emacs-w3m from other application programs.

Moving in an emacs-w3m buffer won't be painful if you're an Emacs user already, since many of the standard keys work as intended. For instance, *C-n*, *C-v* and *C-s* (which are commonly used to move down one line, one page, or search downwards for a word) are valid keys in an emacs-w3m buffer.

To follow a link, use the ⟨RET⟩ key. You have to move the point to a link to do this; links are easily recognizable in a buffer because they are not displayed like ordinary text: they can be underlined, or have a different color. The face `w3m-anchor` controls how they are to be displayed (see Chapter 5 [Customizable Variables], page 30).

⟨RET⟩
⟨right⟩ (Lynx-like keymap only)
: Display the page pointed by the link under point (`w3m-view-this-url`).

  The exact behavior of this command depends on the properties of the link under point, and on whether you give it a prefix argument or not. See Chapter 12 [Emacs-w3m Functions], page 95 for details.

As mentioned above, you can be prompted for a URL when you use the command *M-x w3m*, by setting the `w3m-quick-start` variable to `nil`. In an emacs-w3m buffer, there are also two popular ways to go to new pages by entering their URLs, see below.

⟨RET⟩
: In an emacs-w3m buffer, you can be prompted for a URL in the minibuffer and make emacs-w3m display the corresponding page by hitting ⟨RET⟩ after moving the point to the URL displayed in the '`header-line`'. This feature will feel familiar to you if you are used to GUI-based web browsers like Mozilla. Detailed explanations about this can be found in Chapter 12 [Emacs-w3m Functions], page 95 (`w3m-view-this-url`).

g
: Prompt for a URL in the minibuffer and make emacs-w3m display the corresponding page (independently of the position of the point) in an emacs-w3m buffer. This binding will be familiar to you if you already use Gnus or Mew (`w3m-goto-url`).

G
: Prompt for a URL in the minibuffer, and display it in a new session. This function works just like *g* (*M-x w3m-goto-url*)), except that it opens a new session. Unless you are using emacs-w3m on the character terminal, opening a new session means displaying the page in a new tab. For more information about tabs, please refer to Section 3.5 [Using Tabs], page 18 (`w3m-goto-url-new-session`).

c (Lynx-like keymap)
y (Info-like keymap)
: Display the URL of the page being displayed in the echo area and put it in the `kill-ring` (`w3m-print-current-url`).

`u` (Lynx-like keymap)
`Y` (Info-like keymap)
: Display the target URL of the link under point in the echo area and put it in the `kill-ring` (`w3m-print-this-url`).

If the page you're reading is today's news or someone's diary, it may have been updated since you loaded it. You can refresh the page using the following command. (This command can also be useful to force a full redisplay of the page if it looks broken.)

`R`
`C-S-l` (Info-like keymap only)
: Reload the page (`w3m-reload-this-page`).

3.1.3 Moving from place to place in a page

Being able to use familiar Emacs movement bindings while browsing a web page probably fascinates you already. Believe it or not, there is more! Since we value your time, we have added keys to move the point, scroll the page or find links in a very fast way. When you master them, you will understand how handy they are.

The keys are assigned so that scroll commands can be called using shorter keystrokes than standard Emacs key bindings. And since places where you can input text are pretty specific in web pages (see Section 3.8 [Submitting Forms], page 21), most keys have special meanings and aren't assigned to `self-insert-command` anymore.

⟨SPC⟩
: Scroll downwards. You may be used to this binding if you use the '`more`' or '`less`' commands, or Emacs's `view-mode` (`w3m-scroll-up-or-next-url`).

⟨DEL⟩
`b`
⟨backspace⟩
`S-`⟨SPC⟩
`C-?`
: Scroll upwards. You may be used to this binding if you use the '`less`' command or Emacs's `view-mode` (`w3m-scroll-down-or-previous-url`).

`>`
: Scroll to the left. The scroll step is given by the `w3m-horizontal-scroll-columns` variable, default 10 (`w3m-scroll-left`).

`<`
: Scroll to the right. The scroll step is given by the `w3m-horizontal-scroll-columns` variable, default 10 (`w3m-scroll-right`).

`.` (Lynx-like keymap)
`S-`⟨left⟩ (Info-like keymap)
: Shift to the left (a fine level horizontal scrolling). The shift step is given by the `w3m-horizontal-shift-columns` variable, default 2 (`w3m-shift-left`).

`,` (Lynx-like keymap)
`S-`⟨right⟩ (Info-like keymap)
: Shift to the right (a fine level horizontal scrolling). The shift step is given by the `w3m-horizontal-shift-columns` variable, default 2 (`w3m-shift-right`).

`M-l`
: Scroll horizontally so that the current position is centered (`w3m-horizontal-recenter`).

The `w3m-mode` major mode defines commands to move to various kinds of things; namely links, forms, and images (whether they are displayed or not).

Let's consider this simple example: suppose we want to search for a word on the widely-known Google search engine. Step one: open `http://www.google.com` in emacs-w3m. Step two: once the page is loaded, hit `]`. Tadaa! The point has moved to the first form input in the page, you can now hit ⟨RET⟩ to enter something in it, and then `C-c C-c` to submit. Without this command, you would have had to move into the page using `C-n`, `C-f` and so forth, it would have been a real pain.

⟨TAB⟩
⟨down⟩ (Lynx-like keymap only)
: Move the point to the next link (an "anchor" in emacs-w3m lingo). More strictly speaking, move the point forwards to the nearest anchor.

`M-`⟨TAB⟩
`S-`⟨TAB⟩
⟨backtab⟩
⟨up⟩ (Lynx-like keymap only)
: Move the point to the previous anchor. More strictly speaking, move the point backwards to the nearest anchor (`w3m-previous-anchor`).

`]`
: Move the point to the next form. More strictly speaking, move the point forwards to the nearest form (`w3m-next-form`).

`[`
: Move the point to the previous form. More strictly speaking, move the point backwards to the nearest form (`w3m-previous-form`).

`}`
: Move the point to the next image. More strictly speaking, move the point forwards to the nearest image (`w3m-next-image`).

`{`
: Move the point to the previous image. More strictly speaking, move the point backwards to the nearest image (`w3m-previous-image`).

3.1.4 Moving from page to page

This section explains how to move from page to page, but not by following links or inputing URLs (these ways of moving are explained in Section 3.1.2 [Launching and Jumping], page 8).

This includes commands to move backwards and forwards in history (the familiar "Back" and "Forward" from other browsers), and the obligatory "Go to the home page" feature. You will probably understand this better after reading the Section 3.3 [Tracing History], page 15 section.

`B` (Lynx-like keymap)
⟨left⟩ (Lynx-like keymap)
`l` (Info-like keymap)
`p` (Info-like keymap)
: Move back one page in history (`w3m-view-previous-page`). With a numeric argument ARG, move back ARG pages. This is the preferred way to go back in time.

`N` (Lynx-like keymap)
`n` (Info-like keymap)
: Move forward one page in history (`w3m-view-next-page`). Of course, this will work only if you've used `B` (`l` for the '`Info-like`' keymap) to move back in history. If called with a numeric argument ARG, move forward ARG pages. This command actually allows you to go back to the future!

`H`
: Move to the home page (`w3m-gohome`). You can specify the URL of the home page by customizing the `w3m-home-page` variable ("about:" by default).

These commands are exclusive features of emacs-w3m, we were able to obtain patents on them, so you won't find them anywhere else. That's why you must learn to use them today! (Just kidding.)

`^`
`u` (Info-like keymap only)
: Attempt to move to the parent directory of the page currently displayed. For instance, it will attempt to move to "http://foo/bar/" when "http://foo/bar/baz" is displayed. This function has been implemented because of the following observation: users of web browsers often have to move up one level of directories because the information they're looking for isn't displayed on the current page (either because it has been deleted, or because a search engine took them to the wrong page).

  When you use another web browser, you usually need to remove the last component from the URL by using the ⟨DEL⟩ key, etc. several times manually. In emacs-w3m, this operation can be performed in only one keystroke, by typing `^`!

  Also, you type this command with prefix as "2 `^`", you visit the upper directory according to input number. you type "0 `^`", you visit the top of this site.

⟨SPC⟩
: When the current point is located at the end of the buffer and you cannot scroll down further, hit the ⟨SPC⟩ key to go on to the "next page" (`w3m-scroll-up-or-next-url`). Here, "next page" means the page referred to in the special "next" header of the current page (read on). This feature has nothing to do with the history.

  You probably noticed that some websites split their contents over several small pages, partly because it's not comfortable for users to wait for huge contents to transfer in one go. For example, search results of search engines often appear like that. On such sites, users often need to follow links manually from one page to the next in order the access all the information.

  Hopefully emacs-w3m attempts to enable you to browse a series of contents over pages as if you were viewing them as a single page, by combining two operations: explicitly following links and scrolling.

  That's how it works: these kinds of websites often have fields such as "next", "prev" or "previous" in the headers of their web pages. These fields contain information about links between divided pages. Emacs-w3m uses these fields to know what to follow.

Let's take an example with Google again to explain how the two operations (following links and scrolling) are combined. Open `http://www.google.com` and search for a word. If you search for a very common word (e.g. "hamburger"), you will get tons of search results. Google returns the first page of search results, after having sorted them using its own algorithm and divided them into several pages. Use ⟨SPC⟩ to scroll through this first page. Keep scrolling until you see the end of the web page in the window. Here, a surprise is waiting for you. If you use a web browser other than emacs-w3m and want to see the rest of the search results, you would have to click on any of the numbers listed under

```
Goooooooogle
1 2 3 4 5 6 7 8...
```

to follow the link. But with emacs-w3m, you can follow the link by just typing ⟨SPC⟩, just as if you were scrolling!

⟨DEL⟩
b
⟨backspace⟩
S-⟨SPC⟩
C-?

When the point is located at the beginning of the buffer and you cannot scroll upward, hit the ⟨DEL⟩ to go to the "previous" page (`w3m-scroll-down-or-previous-url`). Here, the "previous" page means the page which is assigned to the "prev" or "previous" fields in the header of the current page. This function has nothing to do with the history, and works like ⟨SPC⟩ (w3m-scroll-up-or-next-url).

3.1.5 Surfing using the mouse

The emacs-w3m developers went to some trouble to ensure that normal people who aren't Emacs otaku can also use emacs-w3m. You can perform most web browsing operations using only the mouse except for entering text, e.g. URL, forms, etc. Note that sometimes you might still need to use modifier keys since the emacs-w3m developers are all Emacs otaku. . .

Use *mouse-2* to "follow links" (the first basic of web browsing). Under Emacs 22.1 and newer, you can also follow links using *mouse-1*, depending of the value of `mouse-1-click-follows-link`.

mouse-2 Follow the link under the mouse pointer (`w3m-mouse-view-this-url`).

S-⟨mouse-2⟩ Follow the link under the mouse pointer in a new session (`w3m-mouse-view-this-url-new-session`).

Scrollbar, menubar and toolbar are helpful in emacs-w3m when you use a mouse (your Emacs must support them, and you must have enabled them). You can scroll an emacs-w3m window using the scrollbar. You can invoke many emacs-w3m commands described in this manual from the "w3m" menu which appears at the top of the Emacs frame. Note: it's not necessary to use the menubar for most emacs-w3m commands, you can use the toolbar icons instead.

To switch between buffers in an emacs-w3m window using "Tabs" (see Section 3.5 [Using Tabs], page 18), click on the topmost line in an emacs-w3m window directory using *mouse-2* or choose one from the "Tab" menu which appears next but one to the "w3m" menu.

3.1.6 Return to an Ordinary Life

Think back. You probably didn't start Emacs to browse the web, but to, say, replace some editor's built-in interpreter with scheme, write Info documentation or put into print your opinion on software patents. . . who knows? While you were using Emacs, you ran into the need to browse the web for your work. But you happened to be able to see the web page without leaving Emacs at all. Wow.

Now the time has come to return to work. Keep it up or the world won't change! Type *q* if you think you might need to browse the web again. Type *Q* if you don't have any intention to go back to emacs-w3m for a while.

q Close an emacs-w3m window and select the other buffer (`w3m-close-window`).

Q Save the "arrived URLs" list to disk (see Section 3.3 [Tracing History], page 15), save cookies (see Section 5.4 [Cookie Variables], page 43) and really quit emacs-w3m (`w3m-quit`).

3.2 Toggle displaying inline images

If the Emacs version you're using is capable of displaying images in buffers, then emacs-w3m can display them in web pages, just like "graphical" browsers like Mozilla do. You should make sure your Emacs is correctly setup for images before trying to use any of the following commands (see Section 2.1 [Required Emacs Version], page 3).

To toggle displaying of images in the current buffer, use *T* (*I* for the '`Info-like`' keymap). It makes emacs-w3m fetch the images from the server, then display them in the buffer, at the position they would have in a "graphical" browser. If you hit the key again, images will disappear from the buffer.

By default, emacs-w3m won't display images, but you can change its behavior and choose to always display images, for this you need to customize the `w3m-default-display-inline-images` variable and change its value from `nil` to `t`. See Chapter 5 [Customizable Variables], page 30.

Emacs-w3m also comes with nifty features that let you zoom an image in or out, save it to a file, or view it in a external viewer. See also Section 3.1.3 [Moving in a page], page 10 for instructions on how to move from image to image in an emacs-w3m buffer.

T (Lynx-like keymap)
I (Info-like keymap)

Toggle displaying of all the inline images in this buffer (`w3m-toggle-inline-images`). If and only if `transient-mark-mode` is turned on and the region is active, only the images within the region will be turned on.

Note1: whether to display inline images in a page from the start when you first visit the page is controlled by the value of the variable `w3m-default-display-inline-images` (the default is off) as mentioned above. But the visibility of images in pages that you visit from this buffer inherits the last status of the

visibility in this buffer if `w3m-toggle-inline-images-permanently` is non-`nil` (default=`t`). If `w3m-toggle-inline-images-permanently` is `nil`, `w3m-default-display-inline-images` always controls it.

Note2: this command deactivates the region, so you have to set it again if you want to turn on only the images in a certain area again.

t (Lynx-like keymap)
i (Info-like keymap)
: Toggle displaying of the single inline image under the cursor (`w3m-toggle-inline-image`). If and only if `transient-mark-mode` is turned on and the region is active, only the images within the region will be turned on. For the '`Info-like keymap`', this key is bound to the command (`w3m-view-image`) that launches the external viewer if Emacs does not support displaying images.

M-S-t (Lynx-like keymap)
M-S-i (Info-like keymap)
: Turn off displaying of all the inline images in this buffer. (`w3m-turnoff-inline-images`).

I (Lynx-like keymap)
: View the image under point in an external viewer (`w3m-view-image`).

M-i
: Save the image under point to an external file. The default name will be the original name of the image, so most of the time *M-i* ⟨RET⟩ will save the image with the right name (`w3m-save-image`).

M-[
: Zoom out the image under point (`w3m-zoom-out-image`).

M-]
: Zoom in the image under point (`w3m-zoom-in-image`).

3.3 Going back through time and space

Emacs-w3m has several ways to present you with a list of all the pages you visited before. The first way is simply called the "emacs-w3m history", it is a list of the pages you visited in this session, presented hierarchically, that is: when you follow a link, the page you're leaving becomes the "parent" of the page you're going to. It is a very nice to keep track of the pages you visited, and remember from where you came if the history gets too long.

Here is an example of this feature in action, after a short visit to the GNU Project's homepage:

```
GNU's Not Unix! - the GNU Project and the Free Software Foundation (FSF)
  Philosophy of the GNU Project - Free Software Foundation (FSF)
    GNU Emacs - GNU Project - Free Software Foundation (FSF)
     Order from the Free Software Foundation (FSF)
  Links to Other Free Software Sites - GNU Project - Free Software Fo...
    EFF: Homepage
```

(In fact, this example is slightly edited to fit in 72 columns; the URLs won't be cut in the actual emacs-w3m buffer.)

You can get this kind of history using the *s* key (the *o* key for the '`Info-like`' keymap) in any emacs-w3m buffer.

Please note that this history is buffer-local, i.e. specific to an emacs-w3m buffer. But emacs-w3m has a unique feature: when you visit a new page, the history is copied over to the new buffer, so that you can still access the pages you visited so far. This is different from the way Mozilla and others work; in these browsers the history always starts from scratch in new buffers.

Emacs-w3m can do more than just record which pages you visited, it can also save specific locations in those pages, in case you want to go back to the exact same place in the page. Press `C-c C-@`, and the location of the cursor will be stored in history. In order to go back to that particular location within the page, press `C-c C-v` in the emacs-w3m buffer visiting the page.

`s` (Lynx-like keymap)
`C-u s` (Lynx-like keymap)
`o` (Info-like keymap)
`C-u o` (Info-like keymap)
: Display the list of URLs visited in this session. If called with a prefix argument (see below), show the list of arrived URLs instead (`w3m-history`).

`C-c C-@`
`C-c C-`⟨SPC⟩
: Record the position of the cursor in the page in history.

`C-c C-v`
: Move to the position which has been marked with `C-c C-@` (`w3m-history-store-position`) in the currently displayed page.

The other way to have information about past pages is the "arrived URLs" list: it is a list of the last 500 URLs you have visited in emacs-w3m. The list is ordered by date, the most recent coming first, and for each page the time of visit is displayed. Here's an example (edited):

```
Order from the Free Software Foundation (FSF)             22:53:25
GNU Emacs - GNU Project - Free Software Foundation (FS    22:53:05
Philosophy of the GNU Project - Free Software Foundati... 22:52:46
Philosophy of the GNU Project - Free Software Foundati... 22:52:39
EFF: Homepage                                             22:52:18
Links to Other Free Software Sites - GNU Project - Fre... 22:52:07
Links to Other Free Software Sites - GNU Project - Fre... 22:52:07
GNU's Not Unix! - the GNU Project and the Free Softwar... 22:51:32
Bookmarks                                                 22:51:02
The DICT Development Group- upwards                       2003-01-08
the monkey puzzle: new debian packages as an rss feed     2003-01-08
new-debian-packages.rss                                   2003-01-07
it's a miracle                                            2003-01-06
```

You can get this history by passing a prefix argument to the previous command, i.e. using `C-u s` (`C-u o` for the '`Info-like`' keymap). The number of URLs showed in this page is customizable, see the `w3m-keep-arrived-urls` variable. It cannot exceed 500 by default. See Chapter 5 [Customizable Variables], page 30.

Of course, in all cases all the lines showed in the examples are links, you can go to any of the pages you visited previously just like if you were visiting a regular page, by following the link.

Also see the Section 3.1.4 [Moving over pages], page 11 section, it explains how to move in the history with simple keybindings, i.e. the "Back" and "Next" features.

3.4 That's a favorite with me!

Like all modern browsers, emacs-w3m has advanced features related to bookmarks: it lets you classify them in categories, edit them and of course, browse them easily.

3.4.1 Adding a URL to your favorites

There are several ways to add a URL to your bookmarks. The first one is to use the `a` key (or call the `w3m-bookmark-add-current-url` command) to add the page you're currently browsing: it will prompt you for a section to where the bookmark should go (completion is available with the ⟨TAB⟩ key) and will let you edit the title of the bookmark (the default being the title of the current page). Complete these two steps, validating each with ⟨RET⟩, and you will see the message "Added" in the minibuffer, which means (surprise!) that the page has been added to your bookmarks.

Another way to add a bookmark is to use the *M-*`a` key (or call the `w3m-bookmark-add-this-url` command): it adds the URL under point (that means, the URL you would be taken to if you followed the link) to the bookmarks. As before, you will have to input the section for this bookmark and its title, the default being this time the name of the link itself.

The third and final way to do this is to use *C-u* `a`, this time you will be prompted for the URL to add, its section, and the title to use for it in the bookmarks.

`a` Add the current page to the bookmarks, or if called with a prefix argument, prompt for a URL and add it (`w3m-bookmark-add-current-url`).

M-`a` Add the URL under point to the bookmarks (`w3m-bookmark-add-this-url`).

3.4.2 Browse your bookmarks

The easiest way to see the bookmarks is to use the *v* key in an emacs-w3m buffer; another possibility is to go to the special URL `about://bookmark/`. You will see your bookmarks, organized by section, each line being one bookmark. You can browse them exactly like you would browse any other page.

On the bookmarks page a w3m minor mode is activated, the bookmark mode. It adds key bindings to edit the bookmarks. See Section 3.4.3 [Editing Bookmarks], page 17.

v Visit the bookmarks page (`w3m-bookmark-view`).

3.4.3 How to change your bookmarks

The bookmark minor mode (see Section 3.4.2 [Consulting Bookmarks], page 17) offers several key bindings related to bookmark edition, most noticeably *C-k* to kill (i.e. delete) a bookmark, and *E* (*e* for the '`Info-like`' keymap) to edit the bookmark file.

Bookmarks are kept in an HTML file, so you can edit the file by hand, but be very careful: if you erase the comments emacs-w3m needs to recognize section names, things can break easily. If you know the basics of HTML, the file should otherwise be quite self-explanatory.

C-k Kill the bookmark under point (`w3m-bookmark-kill-entry`).

E Visit the bookmarks file (`w3m-bookmark-edit`).

C-_ Undo the last changes (`w3m-bookmark-undo`).

3.5 Everybody likes tabs

Unlike most other text-based browsers, emacs-w3m has support for tabbed browsing. What is tabbed browsing, you might ask? It's very simple: it is a way to represent all active emacs-w3m buffers in a single window, by showing a line at the top which shows all the buffers in a simple and self-explaining way, each buffer being shown as a "tab". This line stays visible all the time and does not scroll with the rest of the buffer, so that you can switch to another buffer, or use the feedback it provides at any moment.

The easiest way to get the feeling of it is to just try, so go on and open an emacs-w3m session. If you didn't change anything to the configuration, the tabs line is active by default, it is this bright line at the top with a smaller rectangle that shows the title of the current page. Now create another w3m buffer (with *G*, for example): now you have two of these rectangles. These are tabs.

The most obvious use of tabs is switching: by clicking with the *mouse-1* button on a tab, you make the buffer it represents active. It also works with the *mouse-2* button, or with rolling the mouse wheel if you are using GNU Emacs. It's a very quick and easy way to work with several emacs-w3m buffers, you just have to point and click, or to roll the mouse wheel. (Yeah yeah, I hear you. You want to switch using the keyboard. Don't worry, it's also possible. It's explained in the next section. Now keep quiet and read on!)

Another nifty feature is the feedback it provides. If you are on a color terminal or window system, emacs-w3m shows the text in the tab in different colors to show the status of the page[1]. For example, when the page is being loaded, the text is in red, and goes back to its default color (usually black) when the loading is complete. This way you can tell with a single glance at the tabs line if the page you're waiting for has arrived or not.

Finally, if the web page provides a favicon, it will be shown in the tab as well[2]. More eye-candy for the emacs-w3m user!

User options:

`w3m-use-tab`
: Whether to activate tabbed browsing or not.

If you are a GNU Emacs user, the mouse wheel allows you not only to go to an adjacent buffer but also to move a buffer to the adjacent place. To do that, press and hold down the control key while you roll the mouse wheel on the tabs line. There are two variables that control how emacs-w3m behaves by the mouse wheel:

`w3m-tab-track-mouse`
: This variable controls whether to make the mouse track the selected tab. The default value is `t`. You may want to set this to `nil` if you use a proportional font for the tab faces. See also `w3m-tab-mouse-position-adjuster`.

[1] Although XEmacs shows all tabs in the same colors at every moment, you can easily distinguish the selected tab and others and see the status of the current page in the modeline.

[2] Under XEmacs, favicons will currently not be shown in the tabs line.

`w3m-tab-mouse-position-adjuster`

This variable contains the values used to adjust the mouse position on tabs when the mouse pointer tracks the selected tab. The default value is `(0.5 . -4)`. It consists of the cons of a floating point number *m* and an integer *n* that are applied to calculating of the mouse position, which is given in pixel units, as follows:

```
(TAB_WIDTH + M) * ORDER + N
```

Where *tab_width* is the pixel width of a tab and *order* is the order number in tabs. The result is rounded towards zero.

Note that the calculation will always fail if you use a proportional font for the tab faces. See also `w3m-tab-track-mouse`.

3.6 Creating, killing and moving across buffers

Sooner or later, you will be addicted to emacs-w3m, and you'll have to manage all your browsing needs with it. To help you with this daunting task, we have imagined many different ways to work with emacs-w3m buffers.

3.6.1 Creating and killing buffers

It is sometimes useful to just create a new buffer without opening a web page in it. This operation is called "creating a twin copy" of a buffer, in emacs-w3m lingo. It will simply create a new buffer whose contents are identical to the currently active buffer.

The opposite of this is closing buffers: you can just close one buffer (because you're not interested in its contents anymore) or you can decide to close all buffers but the current one. Emacs-w3m lets you do this with the following commands:

C-c C-t
M-n Create an identical copy of the currently active buffer, under a new name. This is used to start a new session without loading a web page in the new buffer (`w3m-copy-buffer`).

C-c C-w Close the current emacs-w3m buffer (`w3m-delete-buffer`).

C-c M-w Close all emacs-w3m buffers, but the active one (`w3m-delete-other-buffers`).

3.6.2 Moving across buffers

The commands you will probably use most often are those who allow you to go to an adjacent buffer; that is a buffer just "after" or "before" the current one. The meaning of this will be obvious if you use tabs: the next buffer is the one just after the active one, on the right, and the previous buffer is the one on the left. However, XEmacs displays tabs in random order unfortunately, so you need to pay attention to the number which is displayed in each tab in order to know what is the adjacent buffer if you are using XEmacs. The key bindings for these commands are *C-c C-p* and *C-c C-n*.

These commands understand the numeric argument convention, i.e. if you call them with a number N as argument, you will be taken N buffers away from the current one. For example, to go two buffers on the right from the current position, use *2 C-c C-n*[3].

[3] You can use *C-u 2 C-c C-n* instead of *2 C-c C-n* as usual. But keep in mind all numeric keys and minus-sign are assigned to the numeric prefix arguments in emacs-w3m buffers.

`C-c C-p` Move to the previous emacs-w3m buffer. This is usually the next buffer to the left in the tabs line. If called with a numeric argument N, move N buffers to the previous (`w3m-previous-buffer`).

`C-c C-n` Move to the next emacs-w3m buffer. This is usually the next buffer to the right in the tabs line. If called with a numeric argument N, move N buffers to the next (`w3m-next-buffer`).

If you are a GNU Emacs user, you can also move an emacs-w3m buffer to the adjacent place on the tabs line using the following commands:

`C-c C-.`
`C-c C->` Move the selected emacs-w3m buffer to the right hand adjacent place on the tabs line. If called with a numeric argument N, move N tabs to the right (`w3m-tab-move-right`).

`C-c C-,`
`C-c C-<` Move the selected emacs-w3m buffer to the left hand adjacent place on the tabs line. If called with a numeric argument N, move N tabs to the left (`w3m-tab-move-left`).

Also note that if these commands don't fit you well despite our efforts, you might find what you need in "generalist" buffer management packages such as ibuffer or iswitchb—since emacs-w3m buffers are regular Emacs buffers, they will work fine too.

3.6.3 Selecting buffers from a list

There are two ways to select emacs-w3m buffers from a list. The first one is a minibuffer-based interface, called with `C-c C-a`. You can choose the buffer you want to display using the `M-p` and `M-n` keys (or the ⟨up⟩ and ⟨down⟩ arrow keys), they will make you cycle through the list. You can also edit the prompt and type the title of an existing web page, using ⟨TAB⟩ for completion. For example, if you have a "Google Search" page opened, you can type "Goo" then hit ⟨TAB⟩ and the page title will be completed. After the page name, the buffer name is given (between brackets). Then use ⟨RET⟩ to switch to the buffer you have chosen.

The second and more sophisticated interface is called the emacs-w3m buffer list, it is invoked with `C-c C-s`. It shows you the list of all opened buffers in a separate window (either a vertical or a horizontal window—`C-c C-s` toggles between the two modes) and allows you to view the buffers in real-time: when you move the point in the buffer list, the buffer under point is displayed in the main window, which allows you to have direct visual feedback of the buffer you're switching to.

To move in the buffer list, you can use the `p` and `n` keys (or the arrow keys). In the buffer list, ⟨DEL⟩ and ⟨SPC⟩ allow you to scroll the buffer displayed in the main window, which is handy if you want to check that you're seeing the right buffer. To select the buffer under point, you can use the ⟨RET⟩ key, in which case the buffer list will be buried, or the `w` key, in which case the buffer list will remain visible and the focus given to the main window.

You can also close and create buffers from this menu, using the same bindings as the one used in regular buffers (see Section 3.6.1 [Creating and killing buffers], page 19).

Finally, the `?` key shows a short help, `g` refreshes the list and the `q` key exits the buffer list, not changing the active buffer.

`C-c C-a` Prompt for a buffer name in the minibuffer. `M-p` and `M-n` cycle through the list of existing buffers and ⟨TAB⟩ completes (`w3m-switch-buffer`).

`C-c C-s` Show the buffer list in a separate window (`w3m-select-buffer`). In this window, `C-c C-s` toggles between horizontal and vertical modes, ⟨RET⟩ selects the buffer under point and buries the buffer list, `w` selects the buffer under point and gives it the focus, `n`, `p` and the arrow keys can be used to move down or up.

3.7 Downloading a file

It is possible to download (i.e. fetch, but not display) any web page or file with emacs-w3m: just put the point on the link you want to download and hit `d`. You will be prompted for a filename under which to save the file locally, by default it will be the name of the file on the remote server. Confirm with ⟨RET⟩. The download will be asynchronous and not block your Emacs session, you can continue your emacs-w3m browsing in another buffer if you want.

Please note that this download mechanism uses w3m to download things, you might want to use the more powerful wget downloader instead. Have a look at our friend project "emacs-wget", its homepage is at http://pop-club.hp.infoseek.co.jp/emacs/emacs-wget/.

`d` (Lynx-like keymap)
`D` (Info-like keymap)
Download the file or the page pointed to by the link under point (`w3m-download-this-url`).

`M-d` (Lynx-like keymap)
`d` (Info-like keymap)
Download the contents of URL to a local file (`w3m-download`). You will be prompted for the URL and the name of a local file.

3.8 Filling in HTML forms

These emacs-w3m commands let you move between forms and fill in fields, using simple key bindings and optionally prompting you for values in Emacs windows or in the minibuffer.

The main key binding to remember is ⟨RET⟩. It has different meanings, depending on the thing under point: for textareas, you will be prompted for a value in the minibuffer. For select tags, you will be given a list of choices in an electric Emacs window (you can move using the arrow keys, and pick one with the ⟨RET⟩ key). For radio and checkbox buttons, the ⟨RET⟩ key selects one of the elements.

When in the minibuffer or in the electric window, you can cancel with the `C-c C-q` sequence. To submit the form, use `C-c C-c`.

`]` Jump to the next form (`w3m-next-form`).

`[` Jump to the previous form (`w3m-previous-form`).

`C-c C-c` Submit form at point (`w3m-submit-form`).

⟨RET⟩ Edit the value of the form item under point.

⟨RET⟩ (w3m-form-*-keymap)
Accept the value.

`C-c C-q` (w3m-form-*-keymap)
: Quit editing the form item, leaving changes.

Unless `w3m-form-use-textarea-backup` is set to `nil`, emacs-w3m stores the text you input in textareas in backup files for later reuse. When you start editing a form and there is backup text available, you will be asked whether you want to use it or not. Files to save text are stored in the directory specified by the `w3m-form-textarea-directory` variable.

3.9 Support for web page editing and hacking

For those who usually use Emacs to write documentation or programs, it's very convenient to be able to browse the web in the same Emacs session. For example, if you are editing a HTML file in Emacs, you can preview it without launching an external browser. You can also quickly copy sample code from technical documentation during a programming marathon. . .

How about the opposite? (That is, being able to edit the source of a web page in a web browser.) Wouldn't that be cool? Imagine you found an error in your document after previewing it in emacs-w3m; you probably want to fix it right away. Or if you are a programmer specialized in web technology, sometimes you might want to see the raw HTML file for the current web page. . . especially if you are the author of a Shimbun module (see Section 9.6 [Shimbun Basics], page 84).

It is usually possible to switch to an Emacs buffer visiting an HTML file by using the buffer name, but emacs-w3m adds a specific keybinding for this. Emacs-w3m knows the URL of the web page it is visiting, so why not take advantage of this?

`\`
: Display the current web page in the raw HTML format(`w3m-view-source`).

`=`
: Show the information about currently displayed web page. It includes title, URL, document type, last modified date(`w3m-view-header`).

`E` (Lynx-like keymap)
`e` (Info-like keymap)
: Edit the local file pointed by URL of current page(`w3m-edit-current-url`).

`e` (Lynx-like keymap)
`E` (Info-like keymap)
: Edit the local file pointed by URL under point(`w3m-edit-this-url`).

`M`
: Launch an external browser (other than emacs-w3m) and display the same web page as currently displayed in emacs-w3m(`w3m-view-url-with-external-browser`). The external browser to be used is defined by the variable `w3m-content-type-alist`, depending on the kind of URL.

`|`
: Pipe the source of the web page to a command. You will be prompted for the command (`w3m-pipe-source`).

The (see Chapter 10 [Tips], page 92) section gives more examples on how to integrate emacs-w3m with other commands and Emacs subsystems.

4 Pretty good features

4.1 Convenient ways to search the web

Emacs-w3m comes with advanced features related to search engines, they are accessible through three interfaces:

- The regular search interface, invoked by the `S` key (the `s` key for the '`Info-like`' keymap) in any emacs-w3m buffer. It is a simple interactive way to choose which search engine to use and input a search term; see Section 4.1.1 [The Search Interface], page 23.
- The Quicksearch interface: it is a faster (yet more complicated) way to use search engines, by going to specially crafted URLs. For more information about this feature, see Section 4.1.2 [Quick Searching], page 23.
- The "I'm feeling lucky" feature: if it's enabled (`w3m-enable-google-feeling-lucky`), entering words instead of a regular URL at the URL prompt will begin a Google search for the words automatically, and display the most relevant result. This is useful if you actually want to fetch the most relevant page, it does not display a list of search results.

4.1.1 How to search with emacs-w3m

You can fire up the regular search interface by using the `S` key (the `s` key for the '`Info-like`' keymap) in an emacs-w3m buffer. You will see a prompt in the minibuffer, asking for a search term. Type one or several words at the prompt, then hit ⟨RET⟩. The result page of your search in the engine appears, you can then browse the results, just as if you had used the normal web based entry point to the engine.

You probably noticed that you have not been given a chance to choose which engine you want to search with. By default, emacs-w3m will use the Google search engine, you can change this behavior by customizing the `w3m-search-default-engine` variable (see Chapter 5 [Customizable Variables], page 30), or you can specify the search engine each time you use the command.

To specify which engine to use, you have to give the command a prefix argument (usually, this means hitting `C-u` before the command, e.g. `C-u S` (`C-u s` for the '`Info-like`' keymap). Emacs-w3m will prompt you for an engine, you can choose one by typing its name (completion is also available with the ⟨TAB⟩ key). Once you have made your choice, hit the ⟨RET⟩ key. You can then type your search term, hit ⟨RET⟩, and you will see the search results.

`S` (Lynx-like keymap)
`s` (Info-like keymap)
: Begin a new search. If called with a prefix argument, prompt for the engine to use (`w3m-search`).

4.1.2 An alternative (and fast) way to search the web

If you're a "Web Power User" (and since you're reading this, you probably are), you need a quick and efficient way to perform searches. The Quick Searching feature is one.

What does it do? It lets you launch web searches by simply going to a special URL such as `gg:emacs`. The advantages of this mode of operation are:

- It's fast. You just have to type a URL to choose the engine and the search word(s), in one go.
- It's convenient. With this feature, you can easily open a new emacs-w3m tab or window, and launch a search in it, using for example, the *G* key to open a URL in a new window, and going to a Quicksearch URL. You can also bookmark searches just by bookmarking the special Quicksearch URL.
- It works with the grouping feature. You can launch two searches at the same time, with a URL like `group:gg:emacs&ya:w3m`. This would for instance launch a search for "emacs" on Google and for "w3m" on Yahoo!. See Section 4.2 [Grouping URLs], page 25.

Using it is very simple: suppose you want to search for the word "gnu" on Google and get a list of results. Hit *g* to go to a new URL, and type "gg:gnu". The first part of this expression, "gg" indicates that we want to use the Google search engine. The second term is the word we will be searching for. The prefix and the search term must be separated by a colon. Hit ⟨RET⟩, and you will see the results of your search. Note that you can input several words by separating them with spaces. ⟨SPC⟩ is a self-inserting key in the minibuffer if the "Feeling Lucky" feature is enabled (it is by default; see `w3m-enable-google-feeling-lucky`). If it's disabled, then hit *C-q* first, i.e. *C-q* ⟨SPC⟩.

The default configuration of emacs-w3m includes several prefixes you can use, they are defined in the `w3m-uri-replace-alist` variable. There's for example "gg" for Google, "ggg" for Google Groups, "ya" for Yahoo!, "al" for Altavista, "alc" for Eijirou on the web to name a few. You can also add prefixes for the search engines you define, See Section 4.1.3 [Adding New Search Engines], page 24.

Instead of prefixes, you can also use full engine names in Quicksearch URLs, such as "google" or "yahoo". These names are defined in the `w3m-search-engine-alist` variable.

4.1.3 Using your favorite engines

Emacs-w3m has a number of built-in search engines you can use. What if you want to use your favorite search engine and it's not listed in the known search engines? You have to add it to the list of search engines, and it's quite easy:

1. First, you have to find what's the entry point of the search engine you want to add, for example:

   `http://my.searchengine.com/?query=foobar`

   where foobar is the term you want to search for.

2. Once you have this information, add this to your '`~/.emacs-w3m`' file:

   ```
   (eval-after-load "w3m-search"
     '(add-to-list 'w3m-search-engine-alist
                   '("My engine"
                     "http://my.searchengine.com/?query=%s"
                     nil)))
   ```

   Replace the first field "My engine" with the description of your engine, the second field with the entry point (the '`%s`' is important, it will be replaced by the search term when you issue the search), and the third field is the encoding to use, `nil` or omitting this field means to use the value of `w3m-default-coding-system` as a regular encoding.

For English search engines, you rarely have to worry about this. However, for some Japanese search engines, you may need to specify something (e.g. `euc-japan`) there.

3. You can now use this engine to search, using the normal `S` key (the `s` key for the '`Info-like`' keymap) in emacs-w3m. If you use this engine often, you can also add it to the Quicksearch (see Section 4.1.2 [Quick Searching], page 23) engines and give it a small prefix, by adding this to your '`~/.emacs-w3m`' file instead:

```
(eval-after-load "w3m-search"
  '(progn
     (add-to-list 'w3m-search-engine-alist
                  '("My engine"
                    "http://my.searchengine.com/?query=%s"
                    nil))
     (add-to-list 'w3m-uri-replace-alist
                  '("\\`my:" w3m-search-uri-replace "My engine"))))
```

This way you can also use a URL like `my:foobar` to search for the term "foobar" with your engine.

4.2 Visiting several web pages in one URL

Emacs-w3m can manipulate "group URLs": special URLs that contain several real URLs. When you open these group URLs, emacs-w3m will open one buffer for each URL in the group, allowing you to open several pages in one go.

To build group URLs, you just have to put together (i.e. concatenate) all the addresses you want to open, separating them with the ampersand symbol (that's "&"), and prefixing the grouped URLs with "group:". For example, suppose you want to visit the GNU Project's homepage, `http://www.gnu.org/`, and the Savannah homepage, `http://savannah.nongnu.org/`: the group URL would be

```
group:http://www.gnu.org/&http://savannah.nongnu.org/
```

Since this syntax can be quite hard to use on a daily basis, this feature will be most useful when used with very short URLs (Quicksearch URLs for example, see Section 4.1.2 [Quick Searching], page 23); or in non-interactive contexts.

4.3 It will be fine tomorrow

4.4 Raise your antenna

Antenna is a tool to keep track of changes in web pages. Using Antenna, you can periodically check if particular pages have been updated, and if they haven't, know the last time you saw them.

You can start Antenna using the `A` key in any emacs-w3m buffer. Alternatively, you can go to the special URL `about://antenna/`; it does the same thing.

4.4.1 How to add your web sites to Antenna

If you want to add the visiting web site to Antenna, type the `+` key. You will be taken to the customization buffer of `w3m-antenna-sites`, with all fields already set up for you. You just have to hit the buttons "Save for future sessions" and "Finish".

`+` Add a URL to the Antenna database. If called with a prefix argument, ask for a URL instead of adding the current page (`w3m-antenna-add-current-url`).

4.4.2 Tracking changes with Antenna

On the Antenna page, you will see two sections: one called "Updated" and another called "Visited". In the "Updated" section, you will find websites which have changed since the last Antenna update, and in the "Visited" section, the websites which haven't. In each section, each line stands for one website of the Antenna database, and has the following structure:

'`* 2002/12/15 16:43 (T) My website`'

The first part is the last time the website was updated, or if this information is not available, the last time Antenna noticed a change in this page.

The '`(T)`' stands for "Time", it means that the change was detected because the last modification time of that page has changed since the last Antenna update. Another possible value here is '`S`' (for "Size"), which means that the change has been detected because the size of the page has changed.

The last part of this line is the title you gave to this website when you added it to the database.

Please note that the Antenna database doesn't get automatically updated, you have to update it each time you want to check if the sites have changed, either by hitting `R` in the Antenna page, or by passing a prefix argument to the command (start Antenna with `C-u A`, for example).

If you want to make the Antenna database get updated automatically, set the value of the `w3m-antenna-refresh-interval` variable to a positive integer which is an interval time in seconds.

`A` Visit the Antenna page. If called with a prefix argument, update the Antenna database before displaying it (`w3m-antenna`).

4.5 Showing the tree structure of local directories

Using the `w3m-dtree` command, you can display a tree of all subdirectories of a local directory, and browse it like a regular web page. The emacs-w3m buffer you get when you use this feature is very similar to the output of the external "tree" utility, hence the name. Emacs-w3m adds a bonus: if you call the command with a prefix argument, it will display files as well, turning emacs-w3m into a full-featured file browser.

Here is an example of what an emacs-w3m dtree run looks like:

```
/home/romain/.elisp/emacs-w3m/
 |-CVS/
 |-attic/
 |  +-CVS/
 |-autom4te.cache/
 |-doc/
 |  |-CVS/
 |  +-emacs-w3m/
 |-icons/
```

```
|  +-CVS/
|-patches/
|  +-CVS/
+-shimbun/
   +-CVS/
```

And with a prefix argument, you get something like this instead:

```
/home/romain/.elisp/emacs-w3m/ (allfiles)
 |-(f).cvsignore
 |-(f)BUGS.ja
 |-(f)COPYING
 |-[d]CVS/
 |  |-(f)Entries
 |  |-(f)Repository
 |  +-(f)Root
 |-(f)ChangeLog
 |-(f)ChangeLog.1
 |-(f)Makefile
 |-(f)Makefile.in
 |-(f)README
 |-(f)README.ja
```

D (Lynx-like keymap)
T (Info-like keymap)

Prompt for a local directory in the minibuffer, then display its tree structure. If called with a prefix argument (e.g. *C-u D*, or *C-u T* for the '`Info-like`' keymap), show files in the directories as well (`w3m-dtree`).

4.6 Viewing perl documents

4.7 Searching files with Namazu

(under translation)

4.8 Viewing data in various octal form

(under construction)

These following lines in your '`~/.emacs`' may help you to browse octet data files which are opened with `octet-find-file`.

```
(add-hook 'octet-find-file-hook 'view-mode)
(add-hook 'octet-find-file-hook 'w3m-minor-mode)
```

4.9 Grouping sessions into separate frames

It is possible to manage groups of emacs-w3m sessions in separate frames. One use for this would be to have two emacs-w3m frames, where one contains sessions visiting search engines, and the other sessions visiting news sites.

Emacs-w3m offers some convenient features that allow you to visit many web pages at the same time. For instance, you can use tabs (see Section 3.5 [Using Tabs], page 18) to

visit many pages in new sessions, or do so using a special URL beginning with '`group:`' (see Section 4.2 [Grouping URLs], page 25). However, you may want to group them into separate frames if there are too many pages. If so, the `w3m-fb-mode` command is for you. Note that you have to set the `w3m-use-tab` variable to non-`nil` (`t` by default) and set the `w3m-pop-up-frames` variable to `nil` (the default) in order to use it (see Section 5.1 [General Variables], page 30).

Typing *M-x w3m-fb-mode* toggles the mode, but you can turn the mode on by giving a positive integer as a prefix argument to the command (zero or less turns it off).

When the `w3m-fb-mode` is turned on, the sessions that you start in the current frame will be associated with only that frame. Other sessions that are opened in other frames will similarly only appear in those frames. In other words, sessions associated with one frame don't appear in other frames. `w3m-fb-mode` doesn't create any new frames, so you need to make them yourself in some way.

4.10 Saving and loading sessions

It is possible to save and load the emacs-w3m sessions sets.

You can save the set of the currently opened sessions for the future use. Just hit *M-S* and name the set.

Then you will ask how to take the saved sessions set back, won't you? Hit *M-s* to open the sessions selection menu. The available command keys include:

⟨RET⟩ Open all the sessions of the selected sessions set.

M-s Open the detail menu for the selected sessions set. You can open the sessions one by one in that menu.

d Delete the selected sessions set or the session.

r Rename the selected sessions set.

s Save all the opened sessions. So does *M-S*.

n Move the cursor to the next sessions set.

p Move the cursor to the previous sessions set.

q Quit the sessions selection menu.

Emacs-w3m saves some sessions automatically. If `w3m-session-deleted-save` is non-`nil`, emacs-w3m saves the closed sessions automatically. This would be helpful for recovering a session that has been closed inadvertently. If `w3m-session-automatic-save` is non-`nil`, emacs-w3m saves the opened sessions automatically when quitting emacs-w3m.

Sometimes you might forget the URLs of the pages you viewed with the interest. Of course emacs-w3m helps you even in such a case. If `w3m-session-load-last-sessions` is non-`nil`, emacs-w3m automatically opens the sessions set viewed last. If it is `ask`, you will be asked whether to take the set back (default `nil`).

You may have had a bad experience with a crash. It makes you disappointing, and makes displayed web pages lost. Emacs-w3m helps you also in such a case. If `w3m-session-crash-recovery` is non-`nil`, emacs-w3m saves displayed sessions set to use for crash recovering

automatically and recovers saved sessions when emacs-w3m (or emacs, etc) crashes (default `t`). If `w3m-session-load-crashed-sessions` is non-`nil`, emacs-w3m automatically recovers the crashed sessions set. If it is `ask`, you will be asked whether to recover the set (default `ask`).

5 Customizable variables

A lot of emacs-w3m variables are customizable via the Custom mechanism, a graphical Emacs interface to define user options. Custom offers several methods to define your customizations, you can use for example *M-x customize-option* for a single option (i.e. an Emacs Lisp variable) or *M-x customize-group* to see all available options (including variables and faces) for a "group" and change them; in which case the group to use is `w3m`.

Alternatively (if you don't want to use Custom), you can put arbitrary Emacs Lisp expressions in your emacs-w3m initialization file, which is '`~/.emacs-w3m`' by default. This example:

```
(setq w3m-home-page "http://emacs-w3m.namazu.org/")
```

would set the default homepage to http://emacs-w3m.namazu.org/. The syntax to use is the same as in your '`~/.emacs`' file. See section "Init File" in *The Emacs Manual*.

Please note that some variables from external modules could be undefined at the time the '`~/.emacs-w3m`' file is loaded, thus making them impossible to modify (of course if you don't care about the default value, you can override them completely in your '`~/.emacs-w3m`') file. The `w3m-search-engine-alist` variable is a typical example (see Section 5.6 [Search Variables], page 44).

`w3m-init-file`
: When emacs-w3m starts, it will read the `w3m-init-file` file. The default value is '`~/.emacs-w3m`'. You probably don't need to change this. This is a normal Emacs Lisp file and can be used to avoid cluttering your '`~/.emacs`' and '`site-init`' files with emacs-w3m stuff. Emacs-w3m will also check for files with the same names as this, but with '`.elc`' and '`.el`' extensions (in other words, '`~/.emacs-w3m.elc`', '`~/.emacs-w3m.el`' and '`~/.emacs-w3m`', in this order).

5.1 General variables

`w3m-accept-languages`
: List of acceptable languages in descending order of priority. The default value is set according to the '`accept_language`' entry of the '`w3m`' configuration file (normally '`~/.w3m/config`').

`w3m-add-referer`
: Rule of sending referers. There are five choices as the valid values of this option.

1. `nil`: this means that emacs-w3m never send referers.
2. `t`: this means that emacs-w3m always send referers.
3. `lambda`: this means that emacs-w3m send referers only when both the current page and the target page are provided by the same server.
4. a cons cell keeping two regular expressions: this means that emacs-w3m send referers when the url of the current page matches the first regular expression and does not match the second regular expression. `Nil` for the regexp matches any url.
5. a function: emacs-w3m send referers when this function which has two arguments, URL and REFERER, returns non-`nil`.

If you're nervous about leaking private WEB browsing history information, set this option to 'nil' or 'lambda'. If your computer belongs to a secret network, you may set a pair of regular expressions to inhibit sending referers which will disclose your private information, as follows:

```
(setq w3m-add-referer
      '("\\`http:\"
        . "\\`http://\\([^./]+\\.\\)*example\\.net/"))
```

`w3m-add-user-agent`
: Non-`nil` means add the User-Agent field to the request header. The value of `w3m-user-agent` is used for the field body.

`w3m-arrived-file`
: Name of the file to keep the arrived URLs database.

`w3m-auto-show`
: Non-`nil` means provide the ability to horizontally scroll the window. Automatic horizontal scrolling happens when the point gets away from both ends of the window, but nothing occurs if `truncate-lines` is set to `nil`.

  This feature works with specific emacs-w3m code; usual `auto-hscroll-mode`, `automatic-hscrolling`, `auto-show-mode` or `hscroll-mode` will all be invalidated in emacs-w3m buffers.

`w3m-charset-coding-system-alist`
: Alist of MIME charsets and coding systems. Both charsets and coding systems must be symbols.

`w3m-coding-system`
: Default coding system used to communicate with the '`w3m`' command.

`w3m-coding-system-priority-list`
: Coding systems in order of priority used for emacs-w3m sessions.

`w3m-command`
: Name of the executable file of the '`w3m`' command. You normally don't have to specify the value, since emacs-w3m looks for the existing commands '`w3m`', '`w3mmee`' and '`w3m-m17n`' (in this order) in the `exec-path` directories in order if it is `nil` in the beginning.

  If you want to use the other '`w3m`' command, specify the value of this variable explicitly in the .emacs file or customize the value and save it. In this case, you need to restart Emacs and emacs-w3m: there is currently no way to apply the changing of the '`w3m`' command to all the emacs-w3m programs safely after loading the '`w3m.elc`' module.

`w3m-command-arguments`
: List of the default arguments passed to the '`w3m`' command. See also `w3m-command-arguments-alist`.

`w3m-command-arguments-alist`
: Alist of regexps matching urls and additional arguments passed to '`w3m`'. A typical usage of this variable is to specify whether to use a proxy server for

particular hosts. The first match made will be used. Here is an example of how to set this variable:

```
(setq w3m-command-arguments-alist
      '(;; Don't use the proxy server to visit local web pages.
        ("^http://\\([^/]*\\.\\)*your-company\\.com\\(/\\|$\\)"
         "-no-proxy")
        ;; Use the proxy server to visit any foreign urls.
        (""
         "-o" "http_proxy=http://proxy.your-company.com:8080/")))
```

Here the first element matches any url where the scheme is '`http`' and the hostname is either '`your-company.com`' or a name ending with '`.your-company.com`'; the proxy server is not used for those hosts. If you are a regexp novice, you can use the `w3m-no-proxy-domains` variable instead.

`w3m-command-environment`
: Alist of environment variables for subprocesses to inherit.

`w3m-confirm-leaving-secure-page`
: If non-`nil`, you'll be asked for confirmation when leaving secure pages. It is STRONGLY recommended to set a non-nil value to this option. You MUST understand what you want to do completely before switching off this option. The default value is `t`.

`w3m-content-type-alist`
: Alist of content types, regexps, commands to view, and filters. Each element is a list which consists of the following data:

  1. Content type.
  2. Regexp matching a url or a file name.
  3. Method to view contents. The following three types may be used:
     a. Lisp function which takes the url to view as an argument.
     b. ("*command*" [*arg*...]) – where "*command*" is the external command and *arg*'s are the arguments passed to the command if any. The symbols `file` and `url` that appear in *arg*'s will be replaced respectively with the name of a temporary file which contains the contents and the string of the url to view.
     c. `nil` which means to download the url into the local file.
  4. Content type that overrides the one specified by `1. Content type`. Valid values include:
     a. Lisp function that takes three arguments *url*, *content-type*, and *charset*, and returns a content type.
     b. String that specifies a content type.
     c. `nil` that means not to override the content type.

`w3m-correct-charset-alist`
: Alist of MIME charsets; strange ones and standard ones.

`w3m-db-history-display-size`
: Maximum number of arrived URLs which are displayed per page.

`w3m-decoder-alist`
: Alist of encoding types, decoder commands, and arguments.

`w3m-default-coding-system`
: Default coding system used to encode url strings and post-data.

`w3m-default-content-type`
: Default value assumed as the content type of local files.

`w3m-default-directory`
: Directory used as the current directory in emacs-w3m buffers. The valid values include a string specifying an existing directory, a symbol of which the value specifies an existing directory, a function which takes a url as an argument and returns a directory, and `nil` (which is the default). If the specified directory does not exist or it is `nil`, the value of `w3m-profile-directory` is used.

  Note that there is an exception: if a page visits a local file or visits a remote file using ftp, the directory in which the file exists is used as the current directory instead.

`w3m-default-save-directory`
: Default directory where downloaded files will be saved to.

`w3m-delete-duplicated-empty-lines`
: Non-`nil` means display two or more continuous empty lines into single.

`w3m-dirlist-cgi-program`
: Name of the CGI program to list a local directory. If it is `nil`, the dirlist.cgi module of the '`w3m`' command will be used.

`w3m-doc-view-content-types`
: List of content types for which to use `doc-view-mode` to view contents. This overrides `w3m-content-type-alist`.

`w3m-edit-function`
: Function used for editing local files. It is used when the `w3m-edit-current-url` command or the `w3m-edit-this-url` command is invoked.

`w3m-edit-function-alist`
: Alist of functions used for editing pages. This option is referred to decide which function should be used to edit a specified page, when either `w3m-edit-current-url` or `w3m-edit-this-url` is invoked. When no suitable function is found from this alist, `w3m-edit-function` is used.

`w3m-enable-google-feeling-lucky`
: Non-`nil` enables you to enter any words as well as a url when prompted. In that case, emacs-w3m uses Google to search for the words. The default value is `t`.

`w3m-encoding-type-alist`
: Alist of file suffixes and content encoding types.

`w3m-file-coding-system`
: Coding system used when writing configuration files. This value will be referred to by the `w3m-save-list` function.

`w3m-file-name-coding-system`
: Coding system used to convert pathnames when emacs-w3m accesses files.

`w3m-fill-column`
: Integer used as the value for `fill-column` in emacs-w3m buffers. If it is positive, pages will be displayed within the columns of that number. If it is zero or negative, the number of columns which subtracted that number from the window width is applied to the maximum width of pages. Note that XEmacs does not always obey this setting.

`w3m-follow-redirection`
: Maximum number of redirections which emacs-w3m honors and follows. If `nil`, redirections are followed by the '`w3m`' command. Don't set it to `nil` if you allow to use cookies (i.e., you have set `w3m-use-cookies` to non-`nil`) since cookies may be shared among many redirected pages.

`w3m-home-page`
: This variable specifies the url string to open when emacs-w3m starts. Don't say HP, it's the abbreviated name of a certain company. ;-)

`w3m-horizontal-scroll-columns`
: Number of steps in columns used when scrolling a window horizontally.

`w3m-horizontal-scroll-division`
: Integer used by the program making the point certainly visible. The cursor definitely does not go missing even when it has been driven out of the window while wandering around anchors and forms in an emacs-w3m buffer.

  Suppose that the value of this variable is N. When the point is outside the left of the window, emacs-w3m scrolls the window so that the point may be displayed on the position within 1/N of the width of the window from the left. Similarly, when the point is outside the right of the window, emacs-w3m scrolls the window so that the point may be displayed on the position of 1/N of the width of the window from the right.

  This feature doesn't work if `w3m-auto-show` is `nil`. The value must be a larger integer than 1.

`w3m-horizontal-shift-columns`
: Number of steps in columns used when shifting a window horizontally. The term '`shifting`' means a fine level scrolling.

`w3m-imitate-widget-button`
: If non-`nil`, imitate the widget buttons on link (anchor) buttons. It is useful for moving about in a Gnus article buffer using ⟨TAB⟩ key. It can also be any Lisp form that should return a boolean value.

`w3m-init-file`
: Your emacs-w3m startup file name. If a file with the '`.el`' or '`.elc`' suffixes exists, it will be read instead.

  Note: This file is used as the startup configuration *NOT* for the '`w3m`' command but for emacs-w3m. In order to modify configurations for the '`w3m`' command, edit the file named '`~/.w3m/config`' normally.

`w3m-input-coding-system`
: Coding system used when writing to '`w3m`' processes. It overrides `coding-system-for-write` if it is not `binary`. Otherwise, the value of the `w3m-current-coding-system` variable is used instead.

`w3m-keep-arrived-urls`
: Maximum number of URLs which the arrived URLs database keeps.

`w3m-keep-cache-size`
: Maximum number of pages to be cached in emacs-w3m.

`w3m-key-binding`
: Type of key binding set used in emacs-w3m sessions. The valid values include `info` which provides '`Info-like`' keys, and `nil` which provides '`Lynx-like`' keys.

`w3m-language`
: Your preferred language used in emacs-w3m sessions.

`w3m-local-directory-view-method`
: Symbol of the method to view a local directory tree. The valid values include `w3m-cgi` using the CGI program specified by the `w3m-dirlist-cgi-program` variable (which see), and `w3m-dtree` using the w3m-dtree Lisp module.

`w3m-local-find-file-function`
: Function used to open local files. If a url of the `file:` scheme in which you entered agrees with the rule of the `w3m-local-find-file-regexps` variable (which see), it is used to open the file.

  Function should take one argument, the string naming the local file. It can also be any Lisp form returning a function. Set this to `nil` if you want to always use emacs-w3m to see local files.

`w3m-local-find-file-regexps`
: `Cons` of two regexps matching and not matching with local file names. If a url of the `file:` scheme in which you entered matches the first form and does not match the latter form, it will be opened by the function specified by the `w3m-local-find-file-function` variable. `Nil` for the regexp matches any file names.

  For instance, the value `(nil . "\\.html?\\'")` allows '`file:///some/where/w3m.el`', not '`file:///any/where/index.html`', to open by the function specified by `w3m-local-find-file-function`. The latter will be opened as a normal web page. Furthermore, if you would like to view some types of contents in the local system using the viewers specified by the `w3m-content-type-alist` variable, you can add regexps matching those file names to the second element of this variable. For example:

```
(setq w3m-local-find-file-regexps
      '(nil . "\\.\\(?:[sx]?html?\\|dvi\\|ps\\|pdf\\)\\'"))
```

  It is effective only when the `w3m-local-find-file-function` variable is set properly.

`w3m-mailto-url-function`
: Function used to handle the `mailto` urls. Function is called with one argument, just a url. If it is `nil`, a function specified by the `mail-user-agent` variable will be used for composing mail messages.

`w3m-mailto-url-popup-function-alist`
: Alist of `(MAJOR-MODE . FUNCTION)` pairs used to pop a mail buffer up. If a user clicks on a `mailto` url and a mail buffer is composed by `mail-user-agent` with the `MAJOR-MODE`, `FUNCTION` will be called with a mail buffer as an argument. Note that the variables `special-display-buffer-names`, `special-display-regexps`, `same-window-buffer-names` and `same-window-regexps` will be bound to `nil` while popping to a buffer up.

`w3m-make-new-session`
: Non-`nil` means making new emacs-w3m buffers when visiting new pages. If it is non-`nil` and there are already emacs-w3m buffers, the `w3m` command makes a new emacs-w3m buffer if a user specifies a url string in the minibuffer, and the `w3m-safe-view-this-url` command also makes a new buffer if a user invokes it in a buffer not being running the `w3m-mode`. The default value is `nil`.

`w3m-mbconv-command`
: Name of the '`mbconv`' command provided by the '`libmoe`' package. The '`libmoe`' package is used when you use the '`w3mmee`' command instead of the '`w3m`' command. See also `w3m-command`.

`w3m-no-proxy-domains`
: List of domain names for which emacs-w3m will not use a proxy server. Each element should be exactly a domain name which means the latter common part of the host names, not a regexp.

`w3m-output-coding-system`
: Coding system used when reading from '`w3m`' processes.

`w3m-pop-up-frames`
: Non-`nil` means pop to a new frame up for an emacs-w3m session. This variable is similar to `pop-up-frames` and does override `w3m-pop-up-windows`. If `w3m-use-tab` is non-`nil` or there is the buffers selection window (for the `w3m-select-buffer` feature), this variable is ignored when creating the second or more emacs-w3m session.

`w3m-pop-up-windows`
: Non-`nil` means split the windows when a new emacs-w3m session is created. This variable is similar to `pop-up-windows` and quite overridden by `w3m-pop-up-frames` as if `pop-up-frames` influences. Furthermore, if `w3m-use-tab` is non-`nil` or there is the buffers selection window (for the `w3m-select-buffer` feature), this variable is ignored when creating the second or more emacs-w3m session.

`w3m-popup-frame-parameters`
: Alist of frame parameters used when creating a new emacs-w3m frame. It allows not only the alist form but also XEmacs' plist form.

`w3m-prefer-cache`
: Non-`nil` means that cached contents are used without checking headers.

`w3m-profile-directory`
: Directory where emacs-w3m config files are loaded from or saved to.

`w3m-quick-start`
: Non-`nil` means let emacs-w3m start quickly w/o requiring confirmation. When you invoke the `w3m` command, it attempts to visit the page of a string like url around the cursor or the value of `w3m-home-page`. You won't be asked for the confirmation then if this value is non-`nil`. Otherwise, you will be prompted for that url with the editing form.

`w3m-redirect-with-get`
: If non-`nil`, use the GET method after redirection. It controls how emacs-w3m works when a server responds the code 301 or 302. Here is an extract from RFC2616:

  Note: RFC 1945 and RFC 2068 specify that the client is not allowed to change the method on the redirected request. However, most existing user agent implementations treat 302 as if it were a 303 response, performing a GET on the Location field-value regardless of the original request method.

`w3m-relationship-estimate-rules`
: Rules to estimate relationships between a retrieved page and others.

`w3m-select-buffer-horizontal-window`
: Non-`nil` means split windows horizontally to open the selection window.

`w3m-select-buffer-window-ratio`
: The percentage of the selection window to the whole frame. The car is used when splitting windows horizontally and the cdr is for splitting windows vertically.

`w3m-show-decoded-url`
: Non-`nil` means show decoded URIs in the echo area, the balloon, etc. This variable can take one of the following five kinds of forms:

  1. t

     Decode URIs using the encoding guessed from the value of `w3m-coding-system-priority-list`.

  2. Coding system

     Decode URIs using this value.

  3. List of coding systems

     Decode URIs using the encoding assumed based on this list.

  4. Alist of predicates and forms described below:

     Each element looks like the `(PREDICATE . ENCODING)` form. `PREDICATE` should be a regexp, a function or a Lisp form, and `ENCODING` should be one of the forms described here excluding this form. If `PREDICATE` is a regexp, it will be tested whether it matches to the target url. If it is a function, it will be called with the target url. If it is a Lisp form, it will be simply

evaluated. Elements are tested in turn until the result of the test of the predicate is true and the encoding which is associated to the predicate is used for decoding URIs.

5. nil

   Don't decode URIs.

`w3m-use-title-buffer-name`
: Non-`nil` means use name of buffer included current title.

`w3m-show-error-information`
: Non-`nil` means show an error information as a web page. Page is made when the foreign server doesn't respond to a request to retrieve data.

`w3m-space-before-favicon`
: String of space char(s) to be put in front of favicon in the mode-line. It may be better to use two or more spaces if you are using oblique or italic font in the modeline.

`w3m-space-before-modeline-icon`
: String of space character(s) to be put in front of the modeline icon. It may be better to use one or more spaces if you are using oblique or italic font in the modeline.

`w3m-terminal-coding-system`
: Default coding system used when writing to '`w3m`' processes. It is just a default value to set process' coding system initially. (This variable name is analogically derived from the behavior of the '`w3m`' command which accepts data from Emacs just like reads from the terminal.)

`w3m-touch-command`
: Name of the executable file of the touch command. Note that the command is required to be able to modify file's timestamp with the '`-t`' option.

`w3m-track-mouse`
: Whether to track the mouse and message the url under the mouse. See also `show-help-function` if you are using GNU Emacs.

  A tip for XEmacs users:

  You can also use the `balloon-help` feature by the *`M-x balloon-help-mode`* command with arg 1. If the window manager decorates the balloon-help frame, and that is not to your taste, you may strip it off with the following directives:

```
For ol[v]wm use this in .Xdefaults:
   olvwm.NoDecor: balloon-help
     or
   olwm.MinimalDecor: balloon-help

For fvwm version 1 use this in your .fvwmrc:
   NoTitle balloon-help
or
   Style "balloon-help" NoTitle, NoHandles, BorderWidth 0
```

```
For twm use this in your .twmrc:
   NoTitle { "balloon-help" }
```

See the ‘balloon-help.el’ file for more information.

`w3m-uri-replace-alist`
: Alist of regexps matching URIs, and some types of replacements. It can be used universally to replace URI strings in the local rule to the valid forms in the Internet.

  Each element looks like the `(REGEXP FUNCTION OPTIONS...)` form. `FUNCTION` takes one or more arguments, a uri and `OPTIONS`. You can use the grouping constructs ‘`\\(...\\)`’ in `REGEXP`, and they can be referred by the ‘`\N`’ forms in a replacement (which is one of `OPTIONS`).

  Here are some predefined functions which can be used for those ways:

  `w3m-pattern-uri-replace`
  : Replace a URI using PATTERN (which is just an `OPTION`). It is allowed that PATTERN contains the ‘`\N`’ forms in the same manner of `replace-match`.

  `w3m-search-uri-replace`
  : Generate valid URLs to query words on some specified search engines. For example, the element

    ```
    ("\\`gg:" w3m-search-uri-replace "google")
    ```

    makes it possible to replace the URI ‘`gg:emacs`’ to a query for the word ‘`emacs`’ on the Google search engine.

`w3m-url-local-directory-alist`
: Alist of URLs and local directories. If directory names of a given URL and the car of an element are the same, emacs-w3m assumes that the file exists in the local directory where the cdr of an element points to. The default value will be set to a value of the `yahtml-path-url-alist` variable which exchanged the car and the cdr in each element if it is available.

`w3m-use-ange-ftp`
: Non-`nil` means that `ange-ftp` or `efs` is used to access FTP servers.

`w3m-use-cygdrive`
: If non-`nil`, use the ‘`/cygdrive/`’ rule when performing `expand-file-name`.

`w3m-use-filter`
: Non-`nil` means use filter programs to convert web contents. See also `w3m-filter-rules` (the ‘`w3m-filter.elc`’ module provides it but might have never been loaded. In that case, to see the default value and the documentation of `w3m-filter-rules`, type *`M-x load-library`* ⟨RET⟩ *`w3m-filter`* ⟨RET⟩).

`w3m-use-form`
: Non-`nil` means make it possible to use form extensions. *(EXPERIMENTAL)*

`w3m-submit-form-safety-check`
: Non-`nil` means ask you for confirmation when submitting a form. The default value is `nil`.

`w3m-use-header-line`
: Non-`nil` means display the header line.

`w3m-use-header-line-title`
: Non-`nil` means display the current title at the header line. This variable is effective only when `w3m-use-tab` is `nil`.

`w3m-use-mule-ucs`
: Non-`nil` means use the multi-script support with Mule-UCS.

`w3m-use-refresh`
: Non-`nil` means honor the REFRESH attribute in META tags. Emacs-w3m arbitrarily takes you to a url specified by that attribute. Note that they may be malicious traps.

`w3m-refresh-minimum-interval`
: Minimum seconds to wait for refresh, when visiting a page by history-back or history-next.

`w3m-use-symbol`
: Non-`nil` means replace symbols that the '`<_SYMBOL>`' tags lead into. It is meaningful only when the '`w3m-m17n`' command is used and (X)Emacs handles unicode charsets.

`w3m-menu-on-forefront`
: Non-`nil` means place the emacs-w3m menus on the forefront of the menu bar. The default value is `nil`.

`w3m-use-tab`
: Non-`nil` means make emacs-w3m a tab browser. It makes it possible to show all emacs-w3m buffers in a single window with the tabs line, and you can choose one by clicking a mouse on it. See also `w3m-use-tab-menubar`.

`w3m-use-tab-menubar`
: Non-`nil` means use the TAB pull-down menu in the menubar. It makes it possible to show all emacs-w3m buffers in a single window, and you can choose one by clicking a mouse on it. This feature requires that Emacs has been built to be able to display multilingual text in the menubar if you often visit web sites written in non-ascii text. See also `w3m-use-tab`.

`w3m-use-toolbar`
: Non-`nil` activates toolbar of '`w3m`'.

`w3m-user-agent`
: String used for the User-Agent field. See also `w3m-add-user-agent`.

`w3m-new-session-in-background`
: Say whether not to focus on a new tab or a new session in target. It influences only when a new emacs-w3m buffer is created.

`w3m-do-cleanup-temp-files`
: Non-`nil` enables emacs-w3m's auto cleanig forgotten temporary files feature. The default is `nil`.

5.2 Variables related to images

`w3m-default-display-inline-images`
: Non-`nil` means display images inline in emacs-w3m buffers. You can toggle the visibility of images with the `w3m-toggle-inline-images` command. See also `w3m-toggle-inline-images-permanently`.

`w3m-favicon-cache-expire-wait`
: The cache will be expired after specified seconds passed since retrieval. If this variable is `nil`, never expired.

`w3m-favicon-cache-file`
: Filename of saving favicon cache. It defaults to the file named '`.favicon`' under the directory specified by the `w3m-profile-directory` variable.

`w3m-favicon-size`
: Size of favicon. This value is used as geometry argument for `convert`.

`w3m-favicon-type`
: Image type of display favicon.

`w3m-favicon-use-cache-file`
: If non-`nil`, use favicon cache file.

`w3m-favicon-default-background`
: Color name used as transparent color of favicon image. `Nil` means to use the background color of the Emacs frame. The null string `""` is special, that will be replaced with the background color of the header line or the mode line on which the favicon is displayed. Note that this value is effective only with Emacs 22 and greater.

`w3m-icon-directory`
: Directory where emacs-w3m should find icon files.

`w3m-imagick-convert-program`
: Program name of ImageMagick's '`convert`'.

`w3m-treat-image-size`
: Non-`nil` means let the '`w3m`' command mind the ratio of the size of images and text. The default value is `t`.

  If it is non-`nil`, the '`w3m`' command will make a '`halfdump`' which reserves rectangle spaces in which images will be put, and also '`alt`' texts will be truncated or padded with spaces so that their display width will be the same as the width of images.

  See also `w3m-pixels-per-character` and `w3m-pixels-per-line`. Those values will be passed to the '`w3m`' command in order to compute columns and lines which images occupy.

`w3m-pixels-per-character`
: Integer used for the `-ppc` argument of the '`w3m`' command. If `nil`, the width of the default face is used. It is valid only when `w3m-treat-image-size` is non-`nil`. The default value is `nil`. If you want to use emacs-w3m in a character terminal and make `w3m-treat-image-size` effective, you need to set this variable properly.

`w3m-pixels-per-line`
: Integer used for the '`-ppl`' argument of the '`w3m`' command. If `nil`, the height of the default face is used. It is valid only when `w3m-treat-image-size` is non-`nil`. Note that a small value may not induce a good result. The default value is '`64`'. If you want to use emacs-w3m in a character terminal and make `w3m-treat-image-size` effective, you need to set this variable properly.

`w3m-resize-image-scale`
: Number of steps in percent used when resizing images.

`w3m-resize-images`
: If non-`nil`, resize images to the specified width and height.

`w3m-show-graphic-icons-in-header-line`
: Non-`nil` means show graphic status indicators in the header-line. If it is `nil`, also the favicon won't be shown in the header-line even if `w3m-use-favicon` is non-`nil`. This variable is currently meaningless under XEmacs.

`w3m-show-graphic-icons-in-mode-line`
: Non-`nil` means show graphic status indicators in the mode-line. If it is `nil`, also the favicon won't be shown in the mode-line even if `w3m-use-favicon` is non-`nil`.

`w3m-toggle-inline-images-permanently`
: Non-`nil` means let the visibility of images continue permanently. The visibility of images is initialized according to `w3m-default-display-inline-images` at the first time, and except that it may be toggled by the `w3m-toggle-inline-images` command, it does not change hereafter, if it is non-`nil`. Otherwise, whether images are visible is initialized according to `w3m-default-display-inline-images` whenever you visit a new page or reload the current page in an emacs-w3m buffer.

`w3m-use-favicon`
: Non-`nil` means show favicon images if they are available. It will be set to `nil` automatically if ImageMagick's `convert` program does not support the ico format.

`w3m-image-default-background`
: Color name used as transparent color of image. `Nil` means to use the background color of the Emacs frame. The null string `""` is special, that will be replaced with the background color of the buffer. Note that this value is effective only with Emacs 22 and greater.

5.3 Variables related to forms

`w3m-form-input-map-buffer-lines`
: Buffer lines for form select map buffer.

`w3m-form-input-select-buffer-lines`
: Buffer lines for form select buffer.

`w3m-form-input-textarea-buffer-lines`
: Buffer lines for form textarea buffer.

`w3m-form-mouse-face`
: Mouse face to highlight selected value.

`w3m-form-treat-textarea-size`
: Non-`nil` means to process textarea size (treat textarea rows).

`w3m-form-use-fancy-faces`
: Use fancy faces to fontify '`<form>`' tags.

`w3m-form-use-textarea-backup`
: Non-`nil` means save and restore backup text saved when you last edited this textarea. Files to save text are stored in the directory specified by the `w3m-form-textarea-directory` variable.

5.4 Variables related to cookies

`w3m-cookie-accept-bad-cookies`
: If `nil`, don't accept bad cookies. If `t`, accept bad cookies. If ask, ask user whether accept bad cookies or not.

`w3m-cookie-accept-domains`
: A list of trusted domain name string.

`w3m-cookie-file`
: File in which cookies are kept.

`w3m-cookie-reject-domains`
: A list of untrusted domain name string.

`w3m-use-cookies`
: Non-`nil` means enable emacs-w3m to use cookies. *(EXPERIMENTAL)*

5.5 Variables related to bookmarks

`w3m-bookmark-file`
: Bookmark file of w3m.

`w3m-bookmark-file-coding-system`
: Coding system for a created bookmark file. This option is used when a new bookmark file is created, or when an existing bookmark file includes ASCII characters only. If the coding system which is used to encode your using bookmark file is different from the value of this option, emacs-w3m does not change the encoding of your bookmark file.

`w3m-bookmark-default-section`
: Default section to add new entry.

`w3m-bookmark-menu-open-new-session`
: If non-`nil`, "Bookmark" menu item open new session.

5.6 Variables related to searching the web

`w3m-search-default-engine`
: Name of the default search engine. The default is '`google`'.

`w3m-search-engine-alist`
: An alist of search engines. Each element looks like (*engine action coding post-data*). *engine* is a string, the name of the search engine. *action* is a string, the URL that performs a search. *action* must contain a `"%s"`, which is substituted by a query string. *coding* is optional value which is coding system for query string. *post-data* is optional value which is a string for POST method search engine. If *coding* is omitted, it defaults to `w3m-default-coding-system`.

`w3m-search-word-at-point`
: Non-`nil` means that the word at point is used as an initial string. If `transient-mark-mode`, this option is ignored and the region is used as an initial string. The default is `t`.

`w3m-search-thing-at-point-arg`
: Argument for 'thing-at-point' used in 'w3m-search-read-query'. The default is `word`.

5.7 Variables related to weather information

`w3m-weather-default-area`
: Default region to check weather. The default is the southern part of Kyoto city.

`w3m-weather-filter-functions`
: Filter functions to remove useless tags. The default value is a list that contains the following function symbols in this order:

  `w3m-weather-extract-contents` `w3m-weather-adjust-contents` `w3m-weather-expand-anchors` `w3m-weather-insert-title`

5.8 Variables related to the dtree feature

`w3m-dtree-default-allfiles`
: If non-`nil`, invert the meaning of the prefix argument given to the `w3m-dtree` command, i.e., the command shows not only directories but also files even if you don't give a prefix argument. The default is `nil`.

`w3m-dtree-directory-depth`
: Integer that controls how deep `w3m-dtree` shows subdirectories. If it is `nil`, files in all subdirectories are shown. The default is `8`.

`w3m-dtree-indent-strings`
: Vector containing strings used for the indentation. The default is `["|-" "+-" "| " " "]`.

`w3m-dtree-stop-strings`
: Vector containing strings used to indent directories under which there are subdirectories hidden because of `w3m-dtree-directory-depth`. The default is `["|=" "+="]`.

5.9 Variables related to antenna

`w3m-antenna-file`
: Name of the file containing antenna URLs. The default value is '`~/.w3m/.antenna`', where '`~/.w3m`' is the default value of `w3m-profile-directory` (see Section 5.1 [General Variables], page 30).

`w3m-antenna-html-skelton`
: Skeleton used for making the html contents of antenna pages.

`w3m-antenna-make-summary-function`
: Function used to make the summary of the site information. The default is `w3m-antenna-make-summary-like-natsumican`. The other ready-made function is `w3m-antenna-make-summary`.

`w3m-antenna-sites`
: List of web sites that `w3m-antenna` watches. The default is `nil`.

`w3m-antenna-sort-changed-sites-function`
: Function used to sort a list of sites having been changed. The default is `w3m-antenna-sort-sites-by-time`. The other ready-made function is `w3m-antenna-sort-sites-by-title`.

`w3m-antenna-sort-unchanged-sites-function`
: Function used to sort a list of sites having not been changed. The default is `w3m-antenna-sort-sites-by-time`. The other ready-made function is `w3m-antenna-sort-sites-by-title`.

5.10 Variables related to perldoc

`w3m-perldoc-command`
: Name of the executable file of '`perldoc`'. The default is `"perldoc"`.

`w3m-perldoc-input-coding-system`
: Coding system used when writing to the '`perldoc`' command. The default value is `euc-japan` if you are in the Japanese language environment. Otherwise it is `utf-8` if it is available, or `iso-latin-1`.

`w3m-perldoc-output-coding-system`
: Coding system used when reading from the '`perldoc`' command. The default is `undecided`.

`w3m-perldoc-pod2html-command`
: Name of the executable file of '`pod2html`'. The default is `"pod2html"`.

`w3m-perldoc-pod2html-arguments`
: Lisp of arguments passed to the '`pod2html`' command. The default is `("--noindex")`.

5.11 Variables related to namazu

`w3m-namazu-command`
: Name of the executable file of Namazu. The default is '`namazu`'.

`w3m-namazu-arguments`
: List of arguments passed to Namazu. The default value is `("-h" "-H" "-n" w3m-namazu-page-max "-w" whence)`. The symbols `w3m-namazu-page-max` and `whence` will be replaced respectively with the value of that variable and a proper value that the program determines properly.

`w3m-namazu-default-index`
: An alias for the default index, or the directory name of it. If this is `nil`, you will be prompted for the directory name whenever you invoke the `w3m-namazu` command with no prefix argument. The default is the value of `namazu-default-dir` if it exists and `namazu-always-query-index-directory` is `nil`. Otherwise `nil`.

`w3m-namazu-index-alist`
: Alist of aliases and index directories. The default value is determined due to `namazu-dir-alist` if any or `nil`.

`w3m-namazu-input-coding-system`
: Coding system used when reading from the namazu process. The default is the value of `namazu-cs-read` if it exists, or `undecided`.

`w3m-namazu-output-coding-system`
: Coding system used when writing to the namazu process. The default is the value of `namazu-cs-write` if it exists, or is determined to `shift_jis-dos` or `euc-japan-unix` due to the system type.

`w3m-namazu-page-max`
: The maximum number of documents retrieved in one search. The default is the value of `namazu-search-num` if any, or `30`.

5.12 Variables related to the octet feature

There is no user option for the moment.

5.13 Variables related to session manager

`w3m-session-file`
: File name to keep sessions.

`w3m-session-time-format`
: Format of saved time.

`w3m-session-automatic-title`
: String of title to save session automatically.

`w3m-session-deleted-title`
: String of title to save session when buffer delete.

`w3m-session-crash-recovery-title`
: String of title to save session to use for crash recovering.

`w3m-session-deleted-keep-number`
: Number to keep sessions when buffers delete.

`w3m-session-automatic-keep-number`
: Number to keep sessions automatically.

`w3m-session-unknown-title`
: String of title to use when title is not specified.

5.14 Hooks

`w3m-after-cursor-move-hook`
: Hook run each time after the cursor moves in emacs-w3m buffers. This hook is called by the `w3m-check-current-position` function by way of `post-command-hook`.

`w3m-delete-buffer-hook`
: Hook run when every emacs-w3m buffer is deleted.

`w3m-display-hook`
: Hook run after displaying pages in emacs-w3m buffers. Each function is called with a url string as the argument. This hook is evaluated by the `w3m-goto-url` function.

`w3m-fontify-after-hook`
: Hook run after fontifying emacs-w3m buffers. This hook is evaluated by the `w3m-fontify` function.

`w3m-fontify-before-hook`
: Hook run when starting to fontify emacs-w3m buffers. This hook is evaluated by the `w3m-fontify` function.

`w3m-form-input-map-mode-hook`
: A hook called after w3m-form-input-map-mode.

`w3m-form-input-map-set-hook`
: A Hook called before w3m-form-input-map-set.

`w3m-form-input-select-mode-hook`
: A hook called after w3m-form-input-select-mode.

`w3m-form-input-select-set-hook`
: A Hook called before w3m-form-input-select-set.

`w3m-form-input-textarea-mode-hook`
: A hook called after w3m-form-input-textarea-mode.

`w3m-form-input-textarea-set-hook`
: A Hook called before w3m-form-input-textarea-set.

`w3m-minor-mode-hook`
: Hook run after `w3m-minor-mode` initialization.

`w3m-mode-hook`
: Hook run after `w3m-mode` initialization. This hook is evaluated by the `w3m-mode` function.

`w3m-select-buffer-hook`
: Hook run when a different emacs-w3m buffer is selected.

`w3m-bookmark-mode-hook`
: Hook run at the end of function 'w3m-bookmark-mode'.

5.15 Other variables

`w3m-async-exec`
: Non-`nil` means execute the '`w3m`' command asynchronously in Emacs process.

`w3m-broken-proxy-cache`
: Set it to `t` if the proxy server seems not to work properly in caching. Note that this may be the double-edged sword; setting it to `t` will likely be harmful if the proxy server sends bad requests (e.g., not including the Host header, see RFC2616 section 14.23) to foreign servers when the '`w3m`' command specifies the '`no-cache`' directive. Also note that it may not be effective if you are using old '`w3m`' command.

`w3m-history-minimize-in-new-session`
: Non-`nil` means minimize copied history so that there's only current page. This variable is effective when creating of the new session by copying (i.e., `w3m-copy-buffer`). The default value is `nil`.

`w3m-history-reuse-history-elements`
: Non-`nil` means reuse the history element when re-visiting the page. Otherwise, a new history element will be created even if there are elements for the same url in the history.

  Emacs-w3m used to operate as the case in which it is non-`nil`, however it sometimes brought about users' dissatisfaction. For example, if a user visited the pages A -> B -> C -> B in order, performing BACK on the second B would let a user visit A. The reason why a user was taken to A rather than C is that the `w3m-history` variable only had the list `(A B C)` as a history and B was the current position at that time.

  The default value for this variable is `nil` which allows the `w3m-history` variable to have the list `(A B C B)`. Where contents of two B's are the identical Lisp objects. So, too much wasting the Lisp resources will be avoided.

  See the documentation for the variables `w3m-history` and `w3m-history-flat` for more information.

`w3m-process-connection-type`
: Value for `process-connection-type` used when communicating with '`w3m`'.

`w3m-process-modeline-format`
: Format used when displaying the progress of the external '`w3m`' process. It shows a percentage of the data loaded from the web server.

`w3m-show-current-title-in-buffer-tab`
: If non-`nil`, show the title strings in the buffers tab. It has no effect if your XEmacs does not support the gutter items.

6 Hooking emacs-w3m into mail/newsreaders

This section introduces three Message User Agents (MUAs). All those MUAs can display HTML mails properly using emacs-w3m. You'll find here HowTo's and some notes about setting up and using emacs-w3m with each of these MUAs.

Quick note about the conventions we use: what does '`message`' mean?
When a Gnus user says '`message`', it often means a draft of a message to be sent as mail or news. However, it is the term used by Mew or Wanderlust users for received mail. They use '`draft`' for the draft of a message to be sent. On the other hand, a received message is called an '`article`' by Gnus users.

6.1 Reading HTML mails in Gnus

First of all, if the user option `mm-text-html-renderer` defaults to `shr` or `gnus-w3m` (i.e., if the function `mm-shr` or `gnus-article-html` is available) and you are satisfied with that function that is Gnus' built-in HTML renderer, you may not want to do anything. See section "Display Customization" in *The Emacs MIME Manual*. Also See section "HTML" in *The Gnus Manual*.

> There's a tip on using `gnus-article-html` under XEmacs in:
> http://article.gmane.org/gmane.emacs.gnus.general/72704

But if Gnus you use is a bit old and neither the `mm-shr` function nor the `gnus-article-html` function is available, or if you'd like to try emacs-w3m to render HTML articles, this section is just for you.

- What can you do with emacs-w3m?

  You can convert HTML spam mails to be human-readable using emacs-w3m. Of course, it works for HTML ham (non-spam) mails as well, and for both emacs-w3m is probably faster than the default converter. You don't need to perform any additional operation. It will simply be displayed.

  On HTML parts of an article buffer, the `w3m-minor-mode` is turned on and you can use the same main keys as the keys of emacs-w3m, for instance, ⟨RET⟩ is for visiting a page which a link in the current position points to. Those keys are defined in the `w3m-minor-mode-command-alist` variable. Keep in mind that some commands are replaced by others similar to them, for security reasons (see below).

- What do you have to do?

  Add this line to your '`~/.gnus.el`' file:

  ```
  (setq mm-text-html-renderer 'w3m)
  ```

  Also put the following line if you want to show images inline in article buffers:

  ```
  (setq mm-inline-text-html-with-images t)
  ```

  If you don't need to use emacs-w3m keys in article buffers, add the following line too:

  ```
  (setq mm-inline-text-html-with-w3m-keymap nil)
  ```

- Notes

  The above description about spam and ham is not for kidding, it's just here to get your attention. Some HTML mails might contain a nasty trick used by spammers, using the '`<img>`' tag which is far more evil than the '`Click Here!`' button. It is most likely

intended to check whether the ominous spam mail has reached your eyes or not, in which case the spammer knows for sure that your email address is valid. It is done by embedding an identifier string into a URL that you might automatically retrieve when displaying the image. If the `mm-w3m-safe-url-regexp` variable has not been changed from the default value, Gnus will never connect to the spammer's site arbitrarily.

You can display images inline in an article buffer if you set `mm-inline-text-html-with-images` to `t`, can't you? No, not exactly: you're still being protected. If you don't care about leaking information (i.e. the fact that your mail address is reachable), set the `mm-w3m-safe-url-regexp` variable to `nil`. The default value for `mm-w3m-safe-url-regexp` is '`"\\`cid:"`' which means we consider that images included in a mail with the '`cid:`' URL are safe (that is, you can display such images without modifying the `mm-w3m-safe-url-regexp` variable).

- Giveaway

  Even when you are in the summary buffer, you can toggle displaying of images in the article buffer. It is effective only when those images are displayed by emacs-w3m, though. Here's an example:

```
(defun gnus-summary-w3m-safe-toggle-inline-images (&optional arg)
  "Toggle displaying of all images in the article buffer.
If the prefix arg is given, force displaying of images."
  (interactive "P")
  (with-current-buffer gnus-article-buffer
    (let ((st (point-min))
          (nd (point-max))
          (w3m-async-exec w3m-async-exec))
      (save-restriction
        (widen)
        (if (or (> st (point-min)) (< nd (point-max)))
            (setq w3m-async-exec nil))
        (article-goto-body)
        (goto-char (or (text-property-not-all (point) (point-max)
                                              'w3m-safe-url-regexp nil)
                       (point)))
        (if (interactive-p)
            (call-interactively 'w3m-toggle-inline-images)
          (w3m-toggle-inline-images arg))))))

(eval-after-load "gnus-sum"
  '(define-key gnus-summary-mode-map
     "vi" 'gnus-summary-w3m-safe-toggle-inline-images))
```

  You can change the key *vi* into something another. Also see section "Summary Buffer" in *The Gnus Manual*.

See also Section 9.1 [Nnshimbun], page 59.

6.2 Reading HTML mails in Mew

By using emacs-w3m with Mew, you can see HTML mails as it intended to be displayed. To do so, put the following line in the '`~/.mew.el`' file:

```
(require 'mew-w3m)
```

With just this, an HTML mail will be displayed in the message window as if it were a plain text. You can still use the *C-c C-e* command (`mew-summary-execute-external`) there.

It is also quite common these days to see mails containing the same information twice, they use the '`multipart/alternative`' format which consists of both a '`text/plain`' part and a '`text/html`' part (what a waste of bandwidth it is). Mew displays only the '`text/plain`' part of such a mail by default. However, you perhaps want to see the '`text/html`' part since you are using emacs-w3m. If so, add the following lines to the '`~/.mew.el`' file:

```
(setq mew-mime-multipart-alternative-list
      '("Text/Html" "Text/Plain" ".*"))
```

There are some customizable variables related to Mew:

`mew-use-w3m-minor-mode`
: If non-`nil`, the `w3m-minor-mode` is turned on in the message buffer where a text/html part is displayed, and you can use the same main keys as the keys of emacs-w3m, for instance, ⟨RET⟩ is for visiting a page which a link in the current position points to. Those keys are defined in the `w3m-minor-mode-command-alist` variable. Keep in mind that some commands are replaced by others similar to them, for security reasons. The default value is `nil`.

`mew-w3m-auto-insert-image`
: If non-`nil`, you can see images inline in the message buffer when you read a multipart/related message. Note that mew-w3m only allows images contained in the message body with a '`cid:`' URL to be displayed (as we consider them safe). The default value is `nil`.

  To activate this feature, add following in your '`~/.mew.el`'.

  ```
  (define-key mew-summary-mode-map "T" 'mew-w3m-view-inline-image)
  ```

  Press "T", toggle the visibility of the images included its message only. Press "C-uT", display the all images included its Text/Html part.

`mew-w3m-cid-retrieve-hook`
: A hook run just after retrieving a '`cid:`' URL. The default value is `nil`.

See also Section 9.2 [Mew Shimbun], page 63.

6.3 Reading HTML mails in SEMI MUAs

You can display HTML mails as human-readable, using emacs-w3m and SEMI MUA, for example, Wanderlust. Since that MUA depends on SEMI (and also FLIM) for MIME functions, we generically call it SEMI MUA. Although SEMI uses Emacs/W3 for rendering HTML mails by default, it can easily be altered to emacs-w3m and it will make your cyber life still more comfortable.

You simply need to put the following line in '`~/.emacs`' file:

```
(require 'mime-w3m)
```

The `mime-w3m` and `mime-w3` modules are functionally alike, as you might have guessed (see how the names sound alike?). The latter is included in the SEMI package.

On HTML parts of an article buffer, the `w3m-minor-mode` is turned on and you can use the same main keys as the keys of emacs-w3m, for instance, ⟨RET⟩ is for visiting a page which a link in the current position points to. Those keys are defined in the `w3m-minor-mode-command-alist` variable. Keep in mind that some commands are replaced by others similar to them, for security reasons.

There are some customizable variables related to the `mime-w3m` module:

`mime-w3m-display-inline-images`
: If it is non-`nil`, images will be displayed inline in HTML mails. If it is the symbol `default` (which is the default) at the first time, the value of this variable will be replaced with the value of the `w3m-default-display-inline-images` variable. You probably don't need to change this.

`mime-w3m-safe-url-regexp`
: Regexp matching URLs which are considered to be safe. The default value is '`"\\`cid:"`' which means we consider that images included in a mail with the '`cid:`' URLs are safe. See also Section 6.1 [Gnus], page 49 about rogue attacks.

`mime-w3m-setup-hook`
: A hook run just after setting up the cooperation of the `mime-w3m` module and SEMI. The default value is `nil`.

By the way, even when you are in the summary buffer, you can toggle displaying of images in the article buffer (which is what is called the message buffer in the Wanderlust community). It is effective only when those images are displayed by emacs-w3m, though. Here's an example for Wanderlust:

```
(defun wl-summary-w3m-safe-toggle-inline-images (&optional arg)
  "Toggle displaying of all images in the message buffer.
If the prefix arg is given, all images are considered to be safe."
  (interactive "P")
  (with-current-buffer wl-message-buffer
    (w3m-toggle-inline-images arg)))

(eval-after-load "wl-summary"
  '(define-key wl-summary-mode-map
     "\M-i" 'wl-summary-w3m-safe-toggle-inline-images))
```

6.4 VM (vieW maiL) is not Wanderlust

The module vm-w3m.el that provides the feature for VM to display html mails and a patch have been handed over to the new VM maintainer, although it has not appeared in the stable version of VM yet. Try visiting the VM home page.

7 There isn't always an answer

7.1 General Questions

- Q. What's emacs-w3m?

  It is an interface program on Emacs which controls w3m. For more information, see Chapter 1 [Introduction], page 2.

- Q. Which emacs versions are supported?

  The following Emacsen have been checked for emacs-w3m support:

  - Emacs 21
  - Emacs 22
  - XEmacs 21.4.17 and later with/without Mule
  - XEmacs 21.5-b19 and later with/without Mule
  - Meadow

  Note that you're required to use APEL if you'd like to run emacs-w3m under XEmacs. For more information, see Section 2.3 [Other Requirements], page 3.

- Q. Which w3m versions are supported?

  The following w3m versions have been checked for emacs-w3m support:

  - w3m-0.3 and later
  - w3mmee-p24-18 + moe-1.5.4

    Note that w3mmee mentioned as the example is configured with the '`lang=many`' option (it can be done by entering 3, when the '`configure`' script prompts you, "Which language do you prefer?"). It also requires the '`libmoe`' package.

- Q. I've already installed APEL in the XEmacs SUMO package, is it ok?

  There are some problems in the XEmacs APEL package (all modules have been compiled for XEmacs with Mule); for instance, the '`std11`' modules conflict with the FLIM's one, etc. Even though you can use '`apel-1.23-pkg.tar.gz`' or later for both XEmacs with Mule and non-Mule XEmacs if you don't use FLIM for the '`shimbun`' features, we recommend you replace it or newly install the original APEL package. See Section 2.3 [Other Requirements], page 3 where to get it from.

- Q. I've gotten the developing version of emacs-w3m with CVS, however I'm missing '`configure`' script.

  It is necessary to run '`autoconf`' first, to generate '`configure`' script.

7.2 Troubleshooting

- Q. Why can't I enter a password on pages which require authentication?

  Make sure the `w3m-async-exec` variable is set to a value other than `nil`.

- Q. Why can't I enter a password for a proxy server which requires authentication?

  Make sure the `w3m-async-exec` variable is set to a value other than `nil`.

- Q. Why can't I follow links?

  Emacs-w3m requires a version of w3m which recognizes the '`-header`' option. Check what version of w3m you use.

- Q. Why do garbage characters appear?

  It could be caused by the following reasons:

  1. Bad HTML file

     If the character set specified by the '`<meta>`' tag differs from the actual contents in an HTML file, it will not be displayed correctly. Use the command *M-x w3m-redisplay-with-charset* ⟨RET⟩ or *C c* to set the correct character set and to force redisplay of the page.

  2. Limitation of the character sets

     A page written by a character set other than ISO-2022-JP(jis), EUC-JP, or SHIFT_JIS may not be displayed correctly. Try one of the following ways:

     a. Any characters defined in Unicode will be displayed correctly if you install the Mule-UCS package (see Section 2.3 [Other Requirements], page 3). You need to make sure that the value of the `w3m-use-mule-ucs` variable is set to a value other than `nil` after installing the Mule-UCS package. In addition, if you want to make Emacs (and also emacs-w3m) handle the character sets EUC-JISX0213 and ISO-2022-JP-3, you have to install the '`jisx0213`' module which is contained in the Mule-UCS package (though it doesn't work under XEmacs unfortunately).

     b. Install w3mmee or w3m-m17n, and set the `w3m-command` variable to the appropriate value. And emacs-w3m will use the multi-lingual features provided by one of those programs. However, a page written by a coding system which Emacs doesn't support may not be displayed correctly. So please install Mule-UCS package if necessary.

  3. Emacsen incompatibility

     Under XEmacs 21.1, pages written by the SHIFT_JIS character set may not be displayed correctly. There is also a problem in XEmacs 21.2 prior to the beta 36 version. You should upgrade your XEmacs if you use such one.

     You should notice that XEmacs versions 21.1 and 21.2 have already been retired officially.

  4. Don't use `standard-display-european`

     It is generally harmful since it often makes Latin characters get displayed incorrectly. For instance, the apostrophe character ('’') which was encoded as '`&#8217`' will be displayed as the character '`u`' with a grave accent as if it had been encoded as '`&#249`'. If the line something like the following is in your '`~/.emacs`' file or site files which Emacs loads when starting up, we strongly recommend you to remove it.

     ```
     (standard-display-european 1)
     ```

     There the argument might be `t`, not `1`.

- Q. Why can't images be shown?

  It could be caused by the following reasons:

  There is a bug in the earlier versions of the '`libungif`' library. You have to install '`libungif-4.1.0b1`' and later.

  You must install the '`gifsicle`' program if you want to run emacs-w3m under XEmacs. See Section 2.3 [Other Requirements], page 3.

Emacs-w3m doesn't support the old versions of w3m. Check what version of w3m you use.

- Q. Why can't I browse pages which require cookies?

  (This is still an experimental feature.)

  Make sure the `w3m-use-cookies` variable is set to a value other than `nil`.

- Q. Why can't I fill in the form?

  (This is still an experimental feature.)

  Make sure the `w3m-use-form` variable is set to a value other than `nil`.

- Q. Why can't I submit a form?

  (This is still an experimental feature.)

  Make sure the `w3m-use-form` variable is set to a value other than `nil`. You also need to use a version of w3m which recognizes the '`-post`' option in order to use this function. Check what version of w3m you use.

- Q. Why are frames not rendered?

  Install w3mmee and put the following line in your '`~/.emacs`' file:

  ```
  (setq w3m-command "w3mmee")
  ```

- Q. Why are favicon images not displayed in the tabs line on GNU Emacs?

  Install the '`convert`' program which is included in ImageMagick. It is available from: http://www.imagemagick.org/

- Q. Why does GNU Emacs get locked when a favicon image is going to be displayed?

- Q. My computer accesses the disk drive violently and says '`process convert exited abnormally with code 10`'. What's the story?

  Do you use an old version of ImageMagick? As far as we know, it happens when you use the '`convert`' program bundled with ImageMagick 5.2.1. It has been confirmed that the '`convert`' program bundled with ImageMagick 5.4.0-5 (and later) works fine.

  If you don't want to use ImageMagick, or if you can't use its most recent version, add the following line in your '`~/.emacs-w3m`' file:

  ```
  (setq w3m-use-favicon nil)
  ```

- Q. Why does not emacs-w3m work with w3mmee?

  If you are using w3mmee configured with the option '`lang=en`' or '`lang=ja`', reconfigure w3mmee with the option '`lang=many`' (it can be done by entering 3, when the '`configure`' script prompts you, "Which language do you prefer?"), and rebuild w3mmee.

- Q. Why I cannot visit web pages using emacs-w3m? There is no problem when visiting local html files or using w3m barefoot, though.

  What is called the asynch patch[1] is applied to the w3m command which some Linux distribution (e.g. Gentoo Linux) contains. It is useful when using w3m barefoot, however it might make emacs-w3m hang. If it is suspected, we recommend you reinstall the w3m command from the original source.

[1] w3m on cygwin

- Q. Why doesn't the emacs-w3m frame pop up to the front?

  It is quite convenient that the *M-x w3m* ⟨RET⟩ command makes the emacs-w3m frame pop to the front even if it is hidden under the other frames. However, it was reported that it does not work when running Emacs which has been built on some platforms (e.g., Fedora Linux) in which the 'metacity' window manager is used. In those systems, other features which raise the Emacs frames will not work, either. If you are in such a miserable circumstance, it might be worth trying the following advice:

```
(if (or (not window-system) (featurep 'xemacs))
    nil
  (defadvice raise-frame
    (after make-it-work (&optional frame) activate)
    "Make it work with the aid of wmctrl."
    (call-process
     "wmctrl" nil nil nil "-i" "-R"
     (frame-parameter (or frame (selected-frame))
                      'outer-window-id))))
```

  Where 'wmctrl' is the external command which you can get from:

  http://sweb.cz/tripie/utils/wmctrl/

  Note that you have to install the 'wmctrl' command before putting the advice into the '~/.emacs' file.

  The following one is currently unnecessary for emacs-w3m, but a certain application needs it to work. (You will lose nothing by adding it if you are in the platform in which the previous one is needed.)

```
(if (or (not window-system) (featurep 'xemacs))
    nil
  (defadvice pop-to-buffer (after enable-it-to-forcus-frame
                                   activate)
    "Enable it to focus frame if `pop-up-frames' is non-nil."
    (when pop-up-frames
      (let ((id (condition-case nil
                    (frame-parameter
                     (window-frame
                      (get-buffer-window (ad-get-arg 0)))
                     'outer-window-id)
                  (error nil))))
        (when id
          (call-process
           "wmctrl" nil nil nil "-i" "-R" id))))))
```

  The last one is perhaps unnecessary but it might be worth trying in some platforms.

```
(if (or (not window-system)(featurep 'xemacs))
    nil
  (defadvice select-frame (around set-input-focus
                                   (frame) activate)
    "Run `select-frame-set-input-focus'."
    (setq ad-return-value (and (framep frame)
```

```
                                (frame-live-p frame)
                                frame))
        (ad-deactivate 'select-frame)
        (unwind-protect
            (select-frame-set-input-focus frame)
          (ad-activate 'select-frame))))
```

These workarounds will become unnecessary in subsequent Emacs releases (22.2 or 23.1).

7.3 Questions of Shimbun Library

- Q. Why are the '`shimbun`' modules not installed?

  Note that the '`shimbun`' modules (files under the '`shimbun/`' directory) won't be installed if the FLIM package has not been installed in your system.

  The '`configure`' script determines automatically whether the FLIM package is installed or not. If the FLIM package is installed in a non-standard directory, the determination fails. In this case, you can use the '`--with-addpath`' configure option to explicitly set the directory name where the FLIM package has been installed. Here's an example:

  ```
  % ./configure --with-addpath=$HOME/share/emacs/site-lisp/flim
  ```

8 You can surely solve it

9 A tool for reading a newspaper

More and more newspapers, mailing list archives, bulletin boards, and individual diaries (such as hyper nikki system, weblogs and blogs) are published on the web. '`Shimbun`' library enables you to read those contents with your favorite mail/news reader. Actually, '`Shimbun`' library provides functions to convert those contents into articles like common e-mails.

> '`Shimbun`' is pronounced "she-n-boon" (but actually vowels shouldn't be prolonged), it means "newspaper" in Japanese.

The '`shimbun`' module has the goal to generate articles that are as readable as normal mail or news posting. This goal is often difficult to achieve as web sites change the html of their articles.

If you notice (even small) annoyances like nonsense images within the text or any other text that is not related to the article please report them using `report-emacs-w3m-bug` (see Chapter 11 [Mailing List], page 94).

'`Shimbun`' library currently supports Asahi Shimbun, Yomiuri On-line, many mailing list archives such as Emacs Devel, XEmacs Beta, Gnus, Mew, and Wanderlust, Slashdot, Slashdot Japan, and a lot of others. For more detail, see Section 9.5 [Shimbun Sites], page 68.

By the way, you have to pay attention to copyright when using '`Shimbun`' library. Copyrights of articles generated with '`Shimbun`' library are reserved by copyright holders of those original contents. Therefore, you are obliged not to violate rights of copyright holders, when you enjoy generated articles. It means that you can enjoy generated articles on "fair use" that is described in the copyright law.

We, emacs-w3m development team, give no warranty to you, if '`Shimbun`' library causes a damage to you, or if you face a lawsuit about violation of copyrights.

'`Shimbun`' library is a collection of many modules, but each of them serves no useful purpose alone. This section explains three typical '`Shimbun`' applications (two of which are included in the '`Shimbun`' library) and how to make '`Shimbun`' modules by yourself (you need to be able to write Emacs Lisp programs).

Note that you need to have installed the FLIM package (and Gnus if you'd like to use '`nnshimbun`') before building and installing emacs-w3m. The FLIM package requires the APEL package. You might also want to see See Section 2.3 [Other Requirements], page 3.

9.1 Turning Gnus into a web browser!

'`Nnshimbun`' is a Gnus back end, but it is distributed with emacs-w3m, not Gnus, exceptionally. '`Nnshimbun`' allows you to turn Gnus into an exceptionally useful web browser. You can skim through the articles on a newspaper's web server without having to see all the advertisement. You can read articles in mailing list archives as if you were subscribed to the list. You can also read submissions in bulletin boards, etc... Note that if you want to followup, you still need to use emacs-w3m as Gnus can't post via the web with '`nnshimbun`'.

See also Section 6.1 [Gnus], page 49 for rendering HTML messages with emacs-w3m if you use Gnus.

The easiest way to get started with '`nnshimbun`' is to use something like the following in the Group buffer:

M-x gnus-group-make-shimbun-group ⟨RET⟩ *asahi* ⟨RET⟩ *national* ⟨RET⟩

Replace '`asahi`' and '`national`' with the keyword corresponding to the server you'd like to connect to and the group you're interested in respectively. You can complete both of those names by using ⟨TAB⟩ or ⟨SPC⟩.

You can also bind that to a key if there's enough room in the `gnus-group-mode-map` map, you can add something like the following in your '`~/.gnus.el`' file:

```
(eval-after-load "gnus-group"
  '(define-key gnus-group-mode-map "Gn"
     'gnus-group-make-shimbun-group))
```

Then, you can use *G n* instead of *M-x gnus-group-make-shimbun-group*.

Could someone book this keystroke at the Gnus Tower?

Besides this, you can use the `gnus-group-make-shimbun-groups` command in order to make all groups for the specified server.

'`Nnshimbun`' simply fetches HTML contents from the web server and displays them as an article, but it will never save articles in the local file system, except if you use persistent articles (see section "Persistent Articles" in *The Gnus Manual*). '`Nnshimbun`' uses NOV files for each '`nnshimbun`' group, and its back end is almost the same as '`nnml`'.

The following '`nnshimbun`' variables can be customized:

`nnshimbun-keep-backlog`
: This variable overrides the `gnus-keep-backlog` variable (see section "Article Backlog" in *The Gnus Manual*) in '`nnshimbun`' groups. If you set `nnshimbun-keep-backlog` to a number *n*, '`nnshimbun`' will store at most *n* old articles in a buffer for later re-fetching. If this variable is non-`nil` and is not a number, '`nnshimbun`' will store *all* read articles (this is not a good idea). The default value is 300.

  Note that smaller values may spoil the `prefetch-articles` feature (see below), since '`nnshimbun`' uses the backlog to keep the prefetched articles.

`nnshimbun-directory`
: Directory where '`nnshimbun`' saves NOV and marks files. The default value is '`~/News/shimbun/`'. This is a server variable (see section "Server Variables" in *The Gnus Manual*).

`nnshimbun-default-group-level`
: The default group level overriding `gnus-level-default-subscribed`. It will be applied to newly created '`nnshimbun`' groups. The default value is `nil`. This is a server variable (see section "Server Variables" in *The Gnus Manual*).

`nnshimbun-marks-is-evil`
: If non-`nil`, Gnus will never generate and use marks file for '`shimbun`' spools. Using marks files makes it possible to backup and restore '`shimbun`' groups separately from '`.newsrc.eld`'. If you have, for some reason, set this to `t`, and want to set it to `nil` again, you should always remove the corresponding marks file (usually named '`.marks`' in the '`shimbun`' group directory, but see `nnshimbun-marks-file-name`) for the group. Then the marks file will be re-generated properly by Gnus. The default value is `nil`. This is a server variable (see section "Server Variables" in *The Gnus Manual*).

You can use the specially made group parameter for 'nnshimbun' in addition to the standard group parameters provided by Gnus[1]. Several parameters for 'nnshimbun' are collected into the single group parameter nnshimbun-group-parameters which is a property list (the values can be different for every group). Here's an example:

```
'(index-range all prefetch-articles off encapsulate-images on
              expiry-wait 6)
```

Below is the documentation for those group parameters and related variables.

prefetch-articles
: In a group where this group parameter is set to something else than off or nil, 'nnshimbun' not only checks for new articles, but also downloads them. Though it will slow checking of new articles down, you won't be kept waiting when reading articles. In the group where this group parameter is not set or its value is nil, the value of the nnshimbun-pre-fetch-article variable (off by default) is used instead.

encapsulate-images
: In a group where this group parameter is set to something else than off or nil, 'nnshimbun' will put image data embedded in the original contents into an article as 'multipart/related' parts of the MIME format. In the group where this group parameter is not set or its value is nil, the value of the nnshimbun-encapsulate-images variable is used instead. The default value for the nnshimbun-encapsulate-images variable is the value of the shimbun-encapsulate-images variable which is provided in the 'shimbun' library (the default value is probably t).

index-range
: You can specify the range of articles to be fetched from the web server using the index-range group parameter. To specify the range, use the following values:

  nil
  all — all pages

  last — only the latest page

  'integer N'
  : the latest N pages

  'Nnshimbun' checks whether there are new articles by parsing the index page of the server. It is possible that there are two or more index pages on the server. For instance, in the case of the mailing list servers, index pages are generally classified according to the date on which the article was posted. It would take a considerable amount of time to check all those huge index pages especially if you are connecting via a slow line.

  If it is possible, 'nnshimbun' won't check index pages which have already been checked at the last connection. If you want to save even more time, use last. It makes 'nnshimbun' refer to only the latest index page for checking new articles.

[1] The easiest way to specify group parameters is to type G c in the group buffer after moving the point to the group you'd like to customize (see section "Group Parameters" in *The Gnus Manual*).

In the group where the `index-range` group parameter is not set or its value is `nil`, the value of the `nnshimbun-index-range` variable (`2` by default) is used.

`nnshimbun-group-parameters-alist`
: This is an Emacs Lisp variable, an alist of regexp of group names and '`nnshimbun`' group parameters. The default value is `nil`. Each element may have the form '`(REGEXP KEYWORD VALUE KEYWORD VALUE...)`', for example:

```
'("^nnshimbun\\+asahi:" index-range all prefetch-articles off
  encapsulate-images on expiry-wait 6)
```

Since you can use this variable to specify the same '`nnshimbun`' group parameters for two or more groups which have similar names (i.e., those groups are likely to have similar characteristics each other), it is useful that it can be used instead of specifying the '`nnshimbun`' group parameters to several groups respectively. If the group parameter has already been set in a group, that takes precedence over this variable.

You can instruct '`nnshimbun`' to expire articles[2]. Keep in mind that when an article is expired, it is not deleted from the remote server, it's still available there. What is deleted is the line in your own NOV file for '`nnshimbun`'[3] corresponding to the article to be expired. Then the article won't appear in the Summary buffer, forever and ever.

If you don't expire articles in '`nnshimbun`' groups, the NOV files will continue to grow fat indefinitely and you may see very old articles in the Summary buffer as if they were existing (in fact, they might have expired three years ago on the remote server!). Even if you try to read such an article, nothing will appear in the article buffer. On the other hand, most mailing list servers generally offer all the past articles. You may not feel like expiring articles in such groups in order to look back with nostalgia to the good old days and to be able to read, eyes filled with tears, articles which you thought long gone.

You can mark '`nnshimbun`' articles as expirable and specify the expiry period in each '`nnshimbun`' group as well as the other mail back ends. However, there are a little differences between '`nnshimbun`' and the other mail back ends:

- First of all, the expiry period is determined with the following priorities. Note that the default value might be different from group to group.
  1. The value of the `expiry-wait` group parameter in a group.
  2. The value produced by evaluating the `nnmail-expiry-wait-function` variable for a group.
  3. The default value provided by the '`shimbun`' module corresponding to a group.
  4. The value of the `nnmail-expiry-wait` variable.
- Second of all, the argument to be passed to the function specified by the `nnmail-expiry-wait-function` variable will contain the names of the back end and the server

[2] There are mainly two ways to expire articles automatically in the '`nnshimbun`' groups. One is to add a group name regular expression (it should begin with "^nnshimbun\\+") to the `gnus-auto-expirable-newsgroups` variable and to put the expiry period for each group into the `nnmail-expiry-wait-function` variable. Another is to set the `auto-expire` group parameter to `t` and to set the expiry period with the `expiry-wait` group parameter in every '`nnshimbun`' group which you want to expire automatically. See See section "Expiring Mail" in *The Gnus manual*, for more information. In the group to which the `expiry-wait` group parameter is not specified, a default expiry period will be applied.

[3] The NOV file for '`nnshimbun`' is named something like "~/News/shimbun/asahi/national/.overview".

like "nnshimbun+asahi:national", while only the group name will be given in the case of the other mail back ends. Here's an example:

```
(setq nnmail-expiry-wait-function
      (lambda (group)
        (cond ((string-equal group "ding") 7)
              ((string-equal group "nnshimbun+ding:ding") 'never))))
```

This means that there are two groups for the same ding mailing list; one is subscribed as a list member, the other is for reading from the mailing list archive at the Gnus Towers. Ahem, isn't it clever? The local mail files in the "ding" group will be expired in seven days and your local disk space will be saved, but you can read even the articles of the last century in the second group (if it is really needed, though).

- Third of all, and this is written down so that you can remember it when you're filled with doubt: even when all articles from a '`nnshimbun`' group should be expired, the most recent one will be kept. This is not to satisfy your indecisive heart, it's because the next time you fetch new articles for this group, '`nnshimbun`' will know where to begin and not fetch all the articles all over again.

The group parameters and the variables related to expiring '`nnshimbun`' articles are:

`expiry-wait`
: Don't be confused, please. The `expiry-wait` group parameter is provided as one of the elements of `nnshimbun-group-parameters`, the specially made group parameter for '`nnshimbun`'. It has the same name and the same meaning as the standard group parameter. You may use whichever you like. If nnshimbun's one is set to non-`nil` value, it takes precedence over the standard one. It is provided in order to concentrate things related to '`nnshimbun`' at one place of the "Gnus Customize" buffer (which will appear by typing *`G c`* in the group buffer) and to realize managing collectively by the `nnshimbun-group-parameters-alist` variable[4]. The values which can be used are a number of expiry period, `never` or `immediate` as well as the standard group parameter.

`nnshimbun-keep-unparsable-dated-articles`
: If this variable is non-`nil`, the articles of which the time of creation (or the time of arrival) is unknown will never be expired, since their age is unknown. The default is `t`. If you set this variable to `nil`, the articles of which the time is unknown will also be expired unconditionally when the time to expire has come. Well, it might prove useful for a general cleaning at the end of a year.

9.2 Reading web newspapers with Mew

Mew Shimbun is an Emacs Lisp program meant to be used with '`shimbun`' and Mew (version 2.1 and later). The '`mew-shimbun`' module will be installed together with emacs-w3m if Mew, APEL, and FLIM are also installed.

We recommend you also see Section 6.2 [Mew], page 50.

[4] We've already prepared the answer to the question that why `auto-expire` etc. aren't included in the nnshimbun's special group parameter? The answer is, `expiry-wait` is handled by the '`nnshimbun`' back end, but `auto-expire` is handled by the Gnus core. Therefore, it is contrary to the design policy of Gnus to extend the Gnus core functions so that it may work for one particular back end (i.e. reading a value from the nnshimbun's special group parameter).

1. Setting things up

   Put the following lines in the last of the '`~/.mew.el`' file:

```
;;; Loading mew-shimbun, defining keys.
;; (setq mew-shimbun-use-unseen t)              ;;;[5]
(require 'mew-shimbun)
(define-key mew-summary-mode-map "G"  (make-sparse-keymap))
(define-key mew-summary-mode-map "Gg" 'mew-shimbun-goto-folder)
(define-key mew-summary-mode-map "GG" 'mew-shimbun-goto-unseen-folder)
(define-key mew-summary-mode-map "Gi" 'mew-shimbun-retrieve)
(define-key mew-summary-mode-map "GI" 'mew-shimbun-retrieve-all)
(define-key mew-summary-mode-map "Gr" 'mew-shimbun-re-retrieve)
(define-key mew-summary-mode-map "GR" 'mew-shimbun-re-retrieve-all)
(define-key mew-summary-mode-map "Ge" 'mew-shimbun-expire)
(define-key mew-summary-mode-map "GE" 'mew-shimbun-expire-all)

;; Specifying 'shimbun' servers and groups to be read with Mew in the
;; mew-shimbun-folder-groups variable.  Each element has the form
;; ("folder" ("server.group" . range) ...).  You can use all,
;; last, and a number for the range item.

(setq mew-shimbun-folder-groups
      '(;; Fetching 'yomiuri.national', 'yomiuri.sports', etc.
        ;; into the '+shimbun/yomiuri' folder collectively.
        ("yomiuri"
         ("yomiuri.national" . 2)
         ("yomiuri.sports". 2)
         ("yomiuri.world". 2))
        ;; Fetching 'security-memo.memo'
        ;; into '+shimbun/security-memo'.
        ("security-memo"
         ("security-memo.memo" . 2))
        ("slashdot-jp"
         ("slashdot-jp.story" . last))
        ;; You can read several groups in one folder ('+shimbun/emacs')
        ;; as follows even if each group comes from a different server.
        ("emacs"
         ("airw.wl" . last)
         ("emacs-w3m.emacs-w3m" . last))
        ;; Fetching diaries into '+shimbun/hns/arisawa' and
        ;; '+shimbun/hns/miyoshi' respectively.
        ("hns/arisawa"
         ("hns.arisawa" . last))
        ("hns/miyoshi"
```

[5] Uncomment this line if you'd like to manage unseen messages. It must be placed before the `(require 'mew-shimbun)` line.

```
("hns.miyoshi" . last))))
```

You did the fundamental setups. For the other user definable variables, use *M-x customize-group* for the `mew-shimbun` group or see the source code.

2. Reading '`shimbun`' messages

   a. Getting started

      Type *G I* (`mew-shimbun-retrieve-all`) first, after setting things up as mentioned above. The '`shimbun`' folders specified by the `mew-shimbun-folder-groups` variable will be created under the '`+shimbun`' parent folder. Typing *G I* is also useful when you have added new groups. You can change the name of the parent folder ('`+shimbun`' by default) by customizing the `mew-shimbun-folder` variable.

   b. Moving into a '`shimbun`' folder

      You can move to any folder (including '`shimbun`') by typing *g* (`mew-summary-goto-folder`), but *G g* (`mew-shimbun-goto-folder`) is restricted to moving to only the '`shimbun`' folder. In addition, folders which have new messages (in other words, folders which have not been scanned) will be displayed when using a prefix argument with *G g* (i.e. *C-u G g*). A prefix argument similarly affects *G G* as well.

   c. Fetching messages in each folder

      You can fetch new messages for the current folder exclusively by typing *G i* (`mew-shimbun-retrieve`) in a '`shimbun`' folder.

   d. Fetching updated messages

      If you perform the *G r* command (`mew-shimbun-re-retrieve`) when a particular message is being displayed, the message will be updated if it is possible, and new messages will be fetched. With a prefix argument (i.e. *C-u G r*), it will attempt to update messages which are marked with the mark specified by the `mew-shimbun-mark-re-retrieve` variable ('`@`' by default). It would be useful for CNET, etc.

      The *G R* command (`mew-shimbun-re-retrieve-all`) checks the freshness of all messages and re-fetches the updated messages. If a prefix argument is given (i.e. *C-u G R*), the messages within the region will be processed. It is probably worthwhile if the site is running the hyper nikki system ('`nikki`' means "diary" in Japanese).

3. Managing unseen messages

   If you have the following setting in the '`~/.mew.el`' file,

   ```
   (setq mew-shimbun-use-unseen t)
   ```

   newly fetched messages will be marked with the mark specified by the `mew-shimbun-mark-unseen` variable ('`*`' by default), and it will disappear automatically after reading the message.

   Normally, the unseen marks will not be saved in the '`.mew-cache`' file if scanning is not performed after adding or deleting marks in the Mew summary mode, but if you add the following line to the '`~/.mew.el`' file,

   ```
   (setq mew-shimbun-use-unseen-cache-save t)
   ```

   the unseen marks will be saved in the '`.mew-cache`' file automatically for the '`shimbun`' groups when exiting Mew or killing a folder (using *C-c C-q*). However, it is not securely

saved[6]. To do this securely, you had better have the habit of performing the '`scan update`' command after reading the folder.

If you don't like the '`*`' mark for unseen messages, customize the `mew-shimbun-mark-unseen` variable (see above). By specifying the mark by *C-u N*, you will be able to lead a better life with taking care of unseen messages.

4. Expiring messages

   You can expire messages if you set the `mew-shimbun-expires` variable beforehand. For example:

```
(setq mew-shimbun-expires
      '(("yomiuri" . 7)
        ("asahi" . 1)
        ("slashdot-jp" . 7)
        ("emacs" . 7)))
```

   If you set this variable as shown above, you can specify the expiry period; 7 days for '`+shimbun/yomiuri`', 1 day for '`+shimbun/asahi`'. Messages in the '`shimbun`' folder where the expiry period is not specified will never be expired. You can use the *G e* command (`mew-shimbun-expire`) to expire the expirable messages in the current folder. The *G E* command (`mew-shimbun-expire-all`) is for expiring the expirable messages in all the '`shimbun`' folders. Note that once the messages have been expired, you cannot recover them.

5. How to mark messages with '`$`' as unseen

   Put the following lines in the '`~/.mew.el`' file in order to define the '`$`' mark and use '`$`' for the mark of unseen. See http://www.mew.org/ml/mew-dist-2.0/msg01251.html if you would like to replace the '`$`' mark with another.

```
;;-----------------------------------------------------------------
;;; Code for using '$' as the unseen mark.
(setq mew-mark-unseen ?$)
(setq mew-shimbun-mark-unseen mew-mark-unseen)
(setq mew-mark-show-list (cons mew-mark-unseen mew-mark-show-list))
(setq mew-mark-afterstep-spec
      (cons (cons mew-mark-unseen '(1 0 1 0 0 0 0))
            mew-mark-afterstep-spec))
(setq mew-mark-spec
      (cons (list mew-mark-unseen "unseen" 0 nil nil nil nil nil)
            mew-mark-spec))
(setq mew-highlight-mark-keywords
      (cons
       (cons mew-mark-unseen 'mew-face-mark-unseen)
       mew-highlight-mark-keywords))
(defface mew-face-mark-unseen
  '((((class color) (type tty))
     (:foreground "green"))
    (((class color) (background light))
```

[6] People who have set the `mew-touch-folder-p` variable to `t` will succeed 100% in saving marks, but people who use `nil` value seem not to be 0% successful.

```
      (:foreground "deep pink" :bold t :italic t))
     (((class color) (background dark))
      (:foreground "thistle"))
     (t (:bold t)))
   "*Face to highlight the unseen mark"
   :group 'mew-highlight)
 (defun mew-summary-unseen (&optional count)
   "Put the unseen mark(default is '$') in COUNT times."
   (interactive "P")
   (mew-mark-put-mark-loop (function mew-summary-unseen-one) count nil))
 (defun mew-summary-unseen-one (&optional no-msg)
   "Put the unseen mark(default is '$') on this message."
   (mew-mark-put-mark mew-mark-unseen no-msg))
 (defun mew-summary-mark-unseen ()
   "Change the '*' mark into the '$' mark."
   (interactive)
   (mew-summary-exchange-mark mew-mark-review mew-mark-unseen))
 (defun mew-thread-mark-unseen ()
   "Put the '$' mark on all messages of the current sub-thread."
   (interactive)
   (mew-thread-mark mew-mark-unseen))

 (define-key mew-summary-mode-map "$"  'mew-summary-unseen)
 (define-key mew-summary-mode-map "m$" 'mew-summary-mark-unseen)
 (define-key mew-summary-mode-map "t$" 'mew-thread-mark-unseen)
 ;;-----------------------------------------------------------------
```

9.3 Reading web newspapers with Wanderlust

Wanderlust includes '`elmo-shimbun`' as an ELMO module, so you can read '`shimbun`' by just accessing a folder beginning with '`@`' (see section "Shimbun Folder" in *The Wanderlust Manual*).

9.4 Using a shell script to fetch shimbun feeds

If you read lots of '`shimbuns`', checking those for new articles can take some time due to emacs-w3m retrieving the feeds one by one. If you want to speed this up, you can use a shell script to retrieve the feeds, which you can either call manually (e.g. from within Emacs) or automatically through schedulers like cron. The feeds must be saved in specially named files, and emacs-w3m will then use those files instead of calling w3m.

The following variables control the local mode:

`shimbun-use-local`
: Setting this to `t` will activate the local mode, meaning that emacs-w3m will first check if a feed is available as a local file. If it cannot be found, it will be retrieved through w3m as usual.

`shimbun-local-path`
: This is the directory where the shimbun files are stored. The default value is `w3m-default-save-directory`.

The file name for a feed is expected to be the MD5 of the URL, truncated to the first 10 characters, appended with the string '`_shimbun`'. You can easily generate the file name for a feed in Emacs through

```
(concat (substring (md5 "http://example/feed") 0 10) "_shimbun")
```

If you use Gnus with '`nnshimbun`', there is already a function which will generate a download shell script for all currently subscribed shimbun groups. Just call `nnshimbun-generate-download-script`, and it will generate the shell script in a new buffer which you can save afterwards. If you call the function with a prefix, it will put an ampersand after each w3m call, so that the feeds are retrieved in parallel.

9.5 Sites supported by Shimbun

This section provides the list of sites supported by '`shimbun`' library. Unfortunately for people who cannot understand Japanese, almost of supported sites are written in Japanese.

9.5.1 Newspapers Supported by Shimbun

These are newspapers that are supported by '`shimbun`' library.

Asahi Shimbun
: asahi.book asahi.book.column asahi.book.news asahi.book.paperback asahi.book.review asahi.book.special asahi.business asahi.car asahi.culture asahi.digital asahi.editorial asahi.edu asahi.english asahi.food asahi.health asahi.housing asahi.igo asahi.international asahi.international.asia asahi.international.column asahi.international.special asahi.international.world asahi.job asahi.kansai asahi.kansai.entertainment asahi.kansai.kokoro asahi.kansai.sumai asahi.kansai.taberu asahi.komimi asahi.life asahi.life.column asahi.national asahi.politics asahi.rss asahi.science asahi.shopping asahi.shopping.column asahi.shopping.yakimono asahi.shougi asahi.sports asahi.sports.baseball asahi.sports.battle asahi.sports.etc asahi.sports.football asahi.sports.golf asahi.sports.rugby asahi.sports.usa asahi.sports.winter asahi.tenjin asahi.travel asahi.wakata

  Those groups generate articles containing only text by default. If you would like to make them generate HTML articles that contain not only text but also photographs, add the following line to your '`~/.emacs`' file:

  ```
  (setq shimbun-asahi-prefer-text-plain nil)
  ```

  On the other hand, you can also use the '`asahi-html`' back end to generate HTML articles. In order to use it, specify '`asahi-html.business`' instead of '`asahi.business`' as the group name for example.

Asahi Shimbun
: asahi-mytown.(hokkaido. . . okinawa)

  The Asahi Shimbun local-news sections including all the prefectures of Japan.

BBC
: bbc.news

Die Welt welt-de.news

Die Zeit zeit-de.auto zeit-de.computer zeit-de.deutschland zeit-de.feuilleton zeit-de.gesundheit zeit-de.international zeit-de.leben zeit-de.literatur zeit-de.musik zeit-de.news zeit-de.reisen zeit-de.schule zeit-de.sport zeit-de.studium zeit-de.wirtschaft zeit-de.wissen zeit-de.zuender

Gendai Net

gendai-net.today gendai-net.syakai gendai-net.sports gendai-net.geino gendai-net.wadai gendai-net.kenko gendai-net.syoku gendai-net.book

Mainichi jp

(This site has been shifted from MSN in October, 2007)

mainichi.flash mainichi.sports mainichi.entertainment mainichi.entertainment.art mainichi.mantan mainichi.electronics mainichi.weekly mainichi.opinion.editorial mainichi.opinion.yoroku mainichi.opinion.hasshinbako mainichi.opinion.eye mainichi.opinion.hito mainichi.opinion.kinji mainichi.opinion.yuraku mainichi.opinion.closeup mainichi.opinion.kaisetsu mainichi.opinion.newsup

Those groups generate HTML articles containing photographs by default. If you would like to make them generate articles that contain only text, add the following line to your '`~/.emacs`' file:

```
(setq shimbun-mainichi-prefer-text-plain t)
```

The New York Times

nytimes.homepage nytimes.news.business nytimes.news.business.media&advertising nytimes.news.business.worldbusiness nytimes.news.business.smallbusiness nytimes.news.business.yourmoney nytimes.news.business.dealbook nytimes.news.education nytimes.news.health nytimes.news.health.policy nytimes.news.health.psychology nytimes.news.world nytimes.news.world.africa nytimes.news.world.americas nytimes.news.world.asia nytimes.news.world.europe nytimes.news.world.middleeast nytimes.news.us nytimes.news.newyork nytimes.news.newyork.thecity nytimes.news.newyork.metro nytimes.news.obituaries nytimes.news.science nytimes.news.science.earth nytimes.news.science.nutrition nytimes.news.science.space nytimes.news.sports nytimes.news.sports.basketball.college nytimes.news.sports.football.college nytimes.news.sports.golf nytimes.news.sports.hockey nytimes.news.sports.other nytimes.news.sports.baseball.pro nytimes.news.sports.basketball.pro nytimes.news.sports.football.pro nytimes.news.sports.soccer nytimes.news.technology nytimes.news.technology.bits nytimes.news.technology.circuits nytimes.news.technology.pogue nytimes.news.washington nytimes.features.arts nytimes.features.arts.design nytimes.features.arts.music nytimes.features.arts.television nytimes.features.automobiles nytimes.features.books nytimes.features.books.review nytimes.features.dining&wine nytimes.features.fashion nytimes.features.fashion.thursdaystyles nytimes.features.fashion.weddings nytimes.features.home&garden nytimes.features.jobs nytimes.features.magazine nytimes.features.movie.news nytimes.features.movie.reviews nytimes.features.realestate nytimes.features.theater nytimes.features.travel nytimes.features.travel.escapes

nytimes.features.week_in_review nytimes.additional.pop_top nytimes.opinion.editorial

The New York Times began to offer news articles for free on September 19, 2007. In spite of having said '`charset=iso-8859-1`', this site often uses the `windows-1252` charset that is a superset of `iso-8859-1`. '`Shimbun`' (and also emacs-w3m) works in even such a case if the `windows-1252` coding system is available in your (X)Emacs.

Nikkan Sports

nikkansports.flash nikkansports.baseball nikkansports.baseball.highschool nikkansports.baseball.amateur nikkansports.baseball.mlb nikkansports.soccer nikkansports.soccer.japan nikkansports.soccer.world nikkansports.sports nikkansports.sumo nikkansports.nba nikkansports.nfl nikkansports.nhl nikkansports.rugby nikkansports.golf nikkansports.motor nikkansports.battle nikkansports.race nikkansports.race.kka nikkansports.entertainment nikkansports.cinema nikkansports.general

Nihon Keizai Shimbun

nikkei.top nikkei.main nikkei.keizai nikkei.sangyo nikkei.tento nikkei.kansai nikkei.it.business nikkei.it.busi_gyoukai nikkei.it.biz-system nikkei.it.sox nikkei.it.data nikkei.it.taidan nikkei.it.internet nikkei.it.broad nikkei.it.net_gyoukai nikkei.it.iptel nikkei.it.tele nikkei.it.broadcast nikkei.it.internet-column nikkei.it.contents nikkei.it.ec nikkei.it.policy nikkei.it.e-gov nikkei.it.mobile nikkei.it.mob_gyoukai nikkei.it.mobsoft nikkei.it.mobcon nikkei.it.money nikkei.it.one nikkei.it.security nikkei.it.net_crime nikkei.it.digital nikkei.it.pc nikkei.kokunai nikkei.markets nikkei.kawase nikkei.kinri nikkei.ft nikkei.dj nikkei.ngyoseki nikkei.gyosuuchi nikkei.gyoseki nikkei.china nikkei.market nikkei.kaigai nikkei.seiji nikkei.shakai nikkei.retto nikkei.sports nikkei.newpro nikkei.release nikkei.release.it.comp nikkei.release.it.peri nikkei.release.it.sys nikkei.release.it.cont nikkei.release.it.net nikkei.release.it.lsi nikkei.release.it.game nikkei.release.it.etc nikkei.release.dist.depart nikkei.release.dist.ryohan nikkei.release.dist.zakka nikkei.release.dist.cosme nikkei.release.dist.car nikkei.release.dist.book nikkei.release.dist.record nikkei.release.dist.food nikkei.release.dist.mercha nikkei.release.dist.mail nikkei.release.dist.netshop nikkei.release.dist.etc nikkei.release.money.bank nikkei.release.money.sec nikkei.release.money.am nikkei.release.money.insu nikkei.release.money.etc nikkei.release.maker.chemi nikkei.release.maker.mecha nikkei.release.maker.car nikkei.release.maker.elec nikkei.release.maker.food nikkei.release.maker.sports nikkei.release.maker.apparel nikkei.release.maker.commu nikkei.release.maker.etc nikkei.release.service.medic nikkei.release.service.rest nikkei.release.service.trans nikkei.release.service.energy nikkei.release.service.enter nikkei.release.service.env nikkei.release.service.consul nikkei.release.service.edu nikkei.release.service.haken nikkei.release.service.life nikkei.release.service.media nikkei.release.service.lease nikkei.release.service.travel nikkei.release.service.etc nikkei.release.const.const nikkei.release.const.house nikkei.release.const.etc nikkei.shasetsu

MSN Sankei News
(This site has been shifted to MSN in October, 2007)

sankei.news.shakai sankei.news.kokusai sankei.news.seiji sankei.news.keizai sankei.news.seikatsu sankei.news.kyouiku sankei.news.sports sankei.news.cutlure sankei.news.chiho sankei.special.komori sankei.special.kuroda sankei.special.ito sankei.special.tamura sankei.special.jieitai sankei.special.kenpo sankei.special.kyouiku sankei.special.kiko sankei.ronsetsu.shucho sankei.ronsetsu.sankeisho sankei.ronsetsu.seiron

Spiegel Online

spiegel.news

Sponichi sponichi.baseball sponichi.soccer sponichi.usa sponichi.others sponichi.society sponichi.entertainment sponichi.horseracing

Sueddeutsche Zeitung

sueddeutsche-de.alles sueddeutsche-de.nachrichten sueddeutsche-de.politik sueddeutsche-de.wirtschaft sueddeutsche-de.finanzen sueddeutsche-de.kino sueddeutsche-de.kultur sueddeutsche-de.sport sueddeutsche-de.muenchen sueddeutsche-de.panorama sueddeutsche-de.leben sueddeutsche-de.gesundheit sueddeutsche-de.computer

Yomiuri Shimbun

yomiuri.atmoney yomiuri.editorial yomiuri.entertainment yomiuri.iryou yomiuri.kyoiku yomiuri.kyoiku.children yomiuri.kyoiku.english yomiuri.kyoiku.qanda yomiuri.kyoiku.renaissance yomiuri.kyoiku.special yomiuri.national yomiuri.politics yomiuri.science yomiuri.sports yomiuri.world

Those groups generate articles containing only text by default. If you would like to make them generate HTML articles that contain not only text but also photographs, add the following line to your '`~/.emacs`' file:

```
(setq shimbun-yomiuri-prefer-text-plain nil)
```

On the other hand, you can also use the '`yomiuri-html`' back end to generate HTML articles. In order to use it, specify '`yomiuri-html.atmoney`' instead of '`yomiuri.atmoney`' as the group name for example.

9.5.2 News Sites Supported by Shimbun

These are news sites that are supported by '`shimbun`' library.

Al Jazeera aljazeera.news aljazeera.africa aljazeera.america aljazeera.asia-pacific aljazeera.central-asia aljazeera.europe aljazeera.middle-east aljazeera.focus aljazeera.business aljazeera.sport aljazeera.programmes

CNET cnet.news cnet.enterprise.software cnet.enterprise.hardware cnet.security cnet.networking cnet.personal.technology cnet.newsmakers cnet.perspectives

CNET Japan

cnet-jp.general cnet-jp.news cnet-jp.special cnet-jp.opinion cnet-jp.blog.geetstate cnet-jp.blog.kenn cnet-jp.blog.lessig cnet-jp.blog.matsumura cnet-jp.blog.nakajima cnet-jp.blog.saeki cnet-jp.blog.sakamoto cnet-jp.blog.sasaki cnet-jp.blog.sentan cnet-jp.blog.staff cnet-jp.blog.takawata cnet-jp.blog.watanabe

CNN Japan
: cnn-jp.business cnn-jp.fringe cnn-jp.science cnn-jp.showbiz cnn-jp.sports cnn-jp.top cnn-jp.usa cnn-jp.world

De-Bug Magazine
: debugmagazin-de.frontpage debugmagazin-de.musik debugmagazin-de.reviews debugmagazin-de.magazin debugmagazin-de.medien debugmagazin-de.podcast debugmagazin-de.musiktechnik debugmagazin-de.screen debugmagazin-de.gadgets debugmagazin-de.games debugmagazin-de.mode

Engadget Japanese
: engadget-ja.top

Excite News
: excite.bit-koneta excite.world-odd

FAU-IAA
: fau.news

Heise Online
: heise.news heise.telepolis

Infoshop News
: infoshop.news

Impress
: impress.enterprise impress.pc impress.dc impress.akiba impress.av impress.game impress.k-tai impress.internet impress.bb impress.forest impress.robot impress.kaden impress.car

ITmedia
: itmedia.news.bursts itmedia.news.domestic itmedia.news.foreign itmedia.news.products itmedia.news.technology itmedia.news.web20 itmedia.news.nettopics itmedia.news.society itmedia.news.security itmedia.news.industry itmedia.news.research itmedia.news.sp_amd itmedia.anchordesk itmedia.bizid itmedia.enterprise itmedia.+D.plusd itmedia.+D.mobile itmedia.+D.pcupdate itmedia.+D.lifestyle itmedia.+D.games itmedia.+D.docomo itmedia.+D.au_kddi itmedia.+D.vodafone itmedia.+D.shopping itmedia.+D.lifestyle.column.asakura itmedia.+D.lifestyle.column.honda itmedia.+D.lifestyle.column.kobayashi itmedia.+D.lifestyle.column.kodera itmedia.+D.lifestyle.column.nishi itmedia.+D.lifestyle.column.ogikubo itmedia.+D.lifestyle.column.tachibana itmedia.+D.lifestyle.column.takemura itmedia.+D.lifestyle.column.unakami

Japan Times
: japantimes.general japantimes.business

LAUT AG
: laut-de.news laut-de.platten laut-de.platten_alternative laut-de.platten_dance laut-de.platten_hiphop platten_jazz laut-de.platten_metal laut-de.platten_pop laut-de.platten_rnb laut-de.platten_rock

N24
: n24-de.boerse n24-de.boulevard n24-de.nachrichten n24-de.netnews n24-de.politik n24-de.sport n24-de.wirtschaft

Open Tech Press

opentechpress-jp.general opentechpress-jp.enterprise opentechpress-jp.opensource opentechpress-jp.security opentechpress-jp.news opentechpress-jp.pr

Perlentaucher

perlentaucher-de.aktuell

Rediff.com

rediff.news

Slashdot slashdot.frontpage slashdot.apple slashdot.askslashdot slashdot.books slashdot.developers slashdot.games slashdot.hardware slashdot.interviews slashdot.IT slashdot.linux slashdot.mobile slashdot.politics slashdot.science

The following variables are available for configuring the comment section of the Slashdot shimbun:

`shimbun-slashdot-get-comments`

If set to `t` (the default), comments will be retrieved for every article. They are separated from the intro text through a formfeed character (i.e. "^L"); you can access them by scrolling the article buffer as usual (for Gnus you can use the "Next page" button and the "Previous page" button). Setting this variable to `nil` will deactivate retrieval of comments.

`shimbun-slashdot-comment-threshold`

Threshold for displayed comments (default: 3). Can be a number between -1 (all comments) and 5 (highest rating).

`shimbun-slashdot-comment-display`

Type of display for the comments (default: "flat"). Can be either "flat", "thread" or "nested". Note that this must be a string, not a symbol.

Slashdot Japan

slashdot-jp.story slashdot-jp.askslashdot slashdot-jp.bookreview slashdot-jp.bsd slashdot-jp.developers slashdot-jp.interview slashdot-jp.linux slashdot-jp.mac slashdot-jp.mobile slashdot-jp.science slashdot-jp.security slashdot-jp.slash slashdot-jp.it slashdot-jp.hardware slashdot-jp.diary.oliver

Add appropriate configurations to the variable `shimbun-slashdot-jp-group-alist`, you can browse other diaries provided at `http://slashdot.jp/`.

Tech-On! by NikkeiBP

tech-on.latestnews tech-on.mobile tech-on.bbint tech-on.d-ce tech-on.AT tech-on.edaonline tech-on.device tech-on.lsi tech-on.silicon tech-on.observer tech-on.fpd tech-on.mono tech-on.embedded tech-on.mecha tech-on.MEMS tech-on.nano tech-on.carele tech-on.board tech-on.mcu tech-on.PLM tech-on.memory tech-on.measurement tech-on.column.mot

Tech-On! is a technology news site brought by NikkeiBP. At least in autumn 2007, it doesn't seem to be, but a login account (that's for free) was needed to read the whole contents of articles formerly. If it becomes required again in

the future, visit the registration page to have it. The following two variables control how you log in:

`shimbun-tech-on-user-name`
: User name to log in on Tech-On! with. If it is `nil`, you will be prompted for a user name when logging in on Tech-On! with. If it is a string, it will be used as a user name and you will never be prompted. If it is neither `nil` nor a string (that is the default), you will never log in.

`shimbun-tech-on-password`
: Password to use to log in on Tech-On! with. If it is `nil`, you will be prompted for a password when logging in on Tech-On! with. If it is a string, it will be used as a password and you will never be prompted. If it is neither `nil` nor a string (that is the default), you will never log in.

Entering them is required only once in the Emacs session at the first time to start reading a Tech-On! article.

HotWired Japan
: wired-jp.news wired-jp.business wired-jp.culture wired-jp.technology wired-jp.blog.ogura wired-jp.blog.sasaki wired-jp.blog.takahashi

Yahoo! Japan
: yahoo.topnews yahoo.news yahoo.politics yahoo.society yahoo.people yahoo.business-all yahoo.market yahoo.stock yahoo.industry yahoo.international yahoo.entertainment yahoo.sports yahoo.computer yahoo.zenkoku yahoo.hokkaido yahoo.aomori yahoo.iwate yahoo.miyagi yahoo.akita yahoo.yamagata yahoo.fukushima yahoo.tokyo yahoo.kanagawa yahoo.chiba yahoo.saitama yahoo.ibaraki yahoo.tochigi yahoo.gunma yahoo.yamanashi yahoo.nagano yahoo.niigata yahoo.toyama yahoo.ishikawa yahoo.fukui yahoo.aichi yahoo.gifu yahoo.shizuoka yahoo.mie yahoo.osaka yahoo.hyogo yahoo.kyoto yahoo.shiga yahoo.nara yahoo.wakayama yahoo.tottori yahoo.shimane yahoo.okayama yahoo.hiroshima yahoo.yamaguchi yahoo.tokushima yahoo.kagawa yahoo.ehime yahoo.kochi yahoo.fukuoka yahoo.saga yahoo.nagasaki yahoo.kumamoto yahoo.oita yahoo.miyazaki yahoo.kagoshima yahoo.okinawa

  The yahoo.news group retrieves the headline news and also the flash news. Those groups generate HTML articles by default. If you would like to make them generate articles containing only text, add the following line to your '`~/.emacs`' file:

```
(setq shimbun-yahoo-prefer-text-plain t)
```

ZDNet Japan
: zdnet-jp.news zdnet-jp.news.network zdnet-jp.news.hardware zdnet-jp.news.software zdnet-jp.news.manage zdnet-jp.news.security zdnet-jp.news.internet zdnet-jp.news.os zdnet-jp.news.db zdnet-jp.news.system zdnet-jp.column zdnet-jp.column.sp1 zdnet-jp.column.netsecurity1 zdnet-jp.column.ea1 zdnet-jp.column.btl zdnet-jp.column.solutionIT

zdnet-jp.channel.security zdnet-jp.channel.ilm zdnet-jp.blog.iida zdnet-jp.blog.mhatta zdnet-jp.blog.kurei zdnet-jp.blog.opensource zdnet-jp.blog.soa zdnet-jp.blog.dp

The Onion
: the-onion.news

9.5.3 Mailing Lists Supported by Shimbun

These are mailing list archives supported by '`shimbun`' library.

Semi-gnus Mailing List in Japan
: airs.semi-gnus-ja

Wanderlust Mailing List
: airs.wl airs.wl-en

Big Brother DataBase Mailing List
: bbdb-ml.bbdb-ml

GNOME Mailing List
: gnome.balsa-list gnome.calendar-list gnome.cvs-commits-list gnome.foundation-announce gnome.foundation-list gnome.fplan-list gnome.gconf-list gnome.gdome gnome.gnome-1.4-list gnome.gnome-announce-list gnome.gnome-components-list gnome.gnome-db-list gnome.gnome-de gnome.gnome-debugger-list gnome.gnome-devel-list gnome.gnome-doc-list gnome.gnome-gui-list gnome.gnome-hackers gnome.gnome-hackers-readonly gnome.gnome-hackers-test gnome.gnome-i18n gnome.gnome-i18n-tools gnome.gnome-kde-list gnome.gnome-list gnome.gnome-office-list gnome.gnome-pilot-list gnome.gnome-sound-list gnome.gnome-themes-list gnome.gnome-ui-hackers gnome.gnome-web-list gnome.gnome-webmaster-list gnome.gnome-workshop-list gnome.gnomecc-list gnome.gnumeric-list gnome.gtk-app-devel-list gnome.gtk-devel-list gnome.gtk-doc-list gnome.gtk-i18n-list gnome.gtk-list gnome.gtk-perl-list gnome.guppi-list gnome.libart gnome.libart-hackers gnome.orbit-list gnome.vote gnome.wm-spec-list gnome.xml gnome.xslt

Java Conference Mailing List
: javaconf.servlet-ml javaconf.business-ml javaconf.duke-in-the-box-ml javaconf.jfriends-ml javaconf.JGT-ml javaconf.jini-ml javaconf.ejb-ml javaconf.cm-ml javaconf.horb-ml javaconf.talk-ml

LinuxCE JP Mailing List
: linuxce-jp.users

Mule Mailing List
: m17n.mule-ja m17n.mule

Meadow Mailing List
: meadow.meadow-develop meadow.meadow-users-jp

Mew Mailing List
: mew.mew-dist mew.mew-win32 mew.mew-int

MagicPoint Mailing List
: mew.mgp-users mew.mgp-users-jp

www.namazu.org Mailing Lists
: namazu.kakasi-commits namazu.kakasi-dev namazu.migemo namazu.namazu-users-en namazu.namazu-users-ja namazu.namazu-devel-ja namazu.namazu-devel-en namazu.namazu-win32-users-ja namazu.sary

emacs-w3m Mailing List
: emacs-w3m.emacs-w3m

NetBSD JP Mailing List
: netbsd.announce-ja netbsd.junk-ja netbsd.tech-misc-ja netbsd.tech-pkg-ja netbsd.port-arm32-ja netbsd.port-hpcmips-ja netbsd.port-mac68k-ja netbsd.port-mips-ja netbsd.port-powerpc-ja netbsd.hpcmips-changes-ja netbsd.members-ja netbsd.admin-ja netbsd.www-changes-ja

Ruby Mailing List
: ruby.comp.lang.ruby ruby.fj.comp.lang.ruby ruby.ruby-dev ruby.ruby-ext ruby.ruby-list ruby.ruby-math ruby.ruby-talk

Toshiba Linux Users JP Mailing List
: toshiba.linux-users-j

w3m-dev Mailing List
: w3m-dev.w3m-dev w3m-dev.w3m-dev-en

digiko Mailing List
: digiko.digiko

XEmacs Mailing List
: xemacs.xemacs-announce xemacs.xemacs-beta-ja xemacs.xemacs-beta xemacs.xemacs-build-reports xemacs.xemacs-cvs xemacs.xemacs-design xemacs.xemacs-mule xemacs.xemacs-nt xemacs.xemacs-patches xemacs.xemacs-users-ja xemacs.xemacs

Security MEMO Mailing List
: security-memo.memo security-memo.free-memo security-memo.social-memo

  Please note that userid and passowrd are required for '`security-memo.*`' so you have to write;

```
machine memo.st.ryukoku.ac.jp
realm input username/password = archives/archives
login archives
passwd archives

machine memo.st.ryukoku.ac.jp
realm input user: archives / password: archives
login archives
passwd archives
```

  in '`~/.w3m/passwd`' and remove group and others access permissions from the file.

Debian JP Mailing List

debian-jp.debian-announce debian-jp.debian-devel debian-jp.debian-www debian-jp.debian-users debian-jp.debian-policy debian-jp.jp-qa

Debian Mailing List

debian.debian-announce debian.debian-commercial debian.debian-firewall debian.debian-french debian.debian-isp debian.debian-italian debian.debian-kde debian.debian-laptop debian.debian-news debian.debian-news-german debian.debian-news-portuguese debian.debian-security-announce debian.debian-testing debian.debian-user debian.debian-user-catalan debian.debian-user-french debian.debian-user-polish debian.debian-user-portuguese debian.debian-user-spanish debian.debian-user-swedish debian.debian-admintool debian.debian-apache debian.debian-autobuild debian.debian-beowulf debian.debian-boot debian.debian-cd debian.debian-ctte debian.debian-debbugs debian.debian-devel debian.debian-devel-announce debian.debian-devel-french debian.debian-devel-games debian.debian-devel-spanish debian.debian-doc debian.debian-dpkg debian.debian-emacsen debian.debian-events-eu debian.debian-events-na debian.debian-faq debian.debian-gcc debian.debian-glibc debian.debian-gtk-gnome debian.debian-hams debian.debian-ipv6 debian.debian-java debian.debian-jr debian.debian-med debian.debian-mentors debian.debian-newmaint debian.debian-newmaint-admin debian.debian-ocaml-maint debian.debian-openoffice debian.debian-perl debian.debian-pilot debian.debian-policy debian.debian-pool debian.debian-python debian.debian-qa debian.debian-qa-private debian.debian-release debian.debian-security debian.debian-snapshots debian.debian-tetex-maint debian.debian-toolchain debian.debian-vote debian.debian-wnpp debian.debian-www debian.debian-x debian.deity debian.debian-chinese debian.debian-chinese-big5 debian.debian-chinese-gb debian.debian-esperanto debian.debian-i18n debian.debian-japanese debian.debian-l10n-catalan debian.debian-l10n-dutch debian.debian-l10n-english debian.debian-l10n-french debian.debian-l10n-italian debian.debian-l10n-portuguese debian.debian-l10n-spanish debian.debian-laespiral debian.debian-russian debian.debian-simplified-chinese debian.debian-68k debian.debian-alpha debian.debian-arm debian.debian-bsd debian.debian-hppa debian.debian-hurd debian.debian-ia64 debian.debian-mips debian.debian-parisc debian.debian-powerpc debian.debian-s390 debian.debian-sparc debian.debian-superh debian.debian-ultralinux debian.debian-win32 debian.debian-all-changes debian.debian-alpha-changes debian.debian-arm-changes debian.debian-books debian.debian-cd-vendors debian.debian-changes debian.debian-consultants debian.debian-curiosa debian.debian-devel-all-changes debian.debian-devel-alpha-changes debian.debian-devel-arm-changes debian.debian-devel-changes debian.debian-devel-hurd-i386-changes debian.debian-devel-i386-changes debian.debian-devel-m68k-changes debian.debian-devel-powerpc-changes debian.debian-devel-sparc-changes debian.debian-hurd-i386-changes debian.debian-i386-changes debian.debian-legal debian.debian-m68k-changes debian.debian-mirrors debian.debian-powerpc-changes debian.debian-project debian.debian-publicity debian.debian-sgml debian.debian-sparc-changes

debian.lcs-eng debian.lsb-confcall debian.lsb-discuss debian.lsb-impl debian.lsb-spec debian.lsb-test debian.spi-announce debian.spi-general debian.vgui-discuss

KDE Mailing List in Japan

kde.Kuser kde.Kdeveloper

Geocrawler

All archives of Geocrawler are supported by '`shimbun`' library. You can use the command *`M-x shimbun-geocrawler-add-group`* ⟨RET⟩, to add your favorite archive to the variable `shimbun-geocrawler-group-alist`.

Mailing list ARChives

Mailing list ARChives (MARC) are supported by '`shimbun`' library. Add a group name of your favorite archive and its URL to the variable `shimbun-marc-aims-group-alist`, and you can browse it.

RedHat Mailing List

redhat.automake redhat.bug-automake redhat.automake-prs redhat.automake-cvs redhat.binutils redhat.binutils-cvs redhat.c++-embedded redhat.crossgcc redhat.cgen redhat.cgen-prs redhat.cgen-cvs redhat.cygwin redhat.cygwin-xfree redhat.cygwin-announce redhat.cygwin-xfree-announce redhat.cygwin-apps redhat.cygwin-patches redhat.cygwin-developers redhat.cygwin-cvs redhat.cygwin-apps-cvs redhat.docbook-tools-discuss redhat.docbook-tools-announce redhat.docbook-tools-cvs redhat.docbook redhat.dssslist redhat.sgml-tools redhat.docbook-apps redhat.ecos-announce redhat.ecos-devel redhat.ecos-discuss redhat.ecos-maintainers redhat.ecos-patches redhat.elix redhat.elix-announce redhat.gdb redhat.gdb-announce redhat.gdb-testers redhat.gdb-testresults redhat.gdb-patches redhat.gdb-cvs redhat.bug-gdb redhat.gdb-prs redhat.libc-alpha redhat.libc-hacker redhat.bug-glibc redhat.glibc-cvs redhat.glibc-linux redhat.bug-gnats redhat.gnats-devel redhat.gnats-announce redhat.gnats-cvs redhat.gsl-discuss redhat.gsl-announce redhat.gsl-cvs redhat.guile redhat.guile-emacs redhat.guile-prs redhat.guile-gtk redhat.bug-guile redhat.guile-cvs redhat.guile-emacs-cvs redhat.insight redhat.insight-announce redhat.insight-prs redhat.installshell redhat.inti redhat.kawa redhat.libffi-discuss redhat.libffi-announce redhat.libstdc++ redhat.libstdc++-cvs redhat.libstdc++-prs redhat.mauve-discuss redhat.mauve-announce redhat.newlib redhat.pthreads-win32 redhat.rhdb redhat.rhdb-announce redhat.rhug-rhats redhat.rpm2html-cvs redhat.rpm2html-prs redhat.rpm2html redhat.sid redhat.sid-announce redhat.sid-cvs redhat.sourcenav redhat.sourcenav-announce redhat.sourcenav-prs redhat.win32-x11 redhat.xconq7 redhat.xconq-announce redhat.xconq-cvs

MacOSX JP Mailing List

macosx-jp.macosx-jp macosx-jp.macosx-dev-jp macosx-jp.macosx-ws-jp macosx-jp.webobjects-jp

SourceForge JP

All archives served by SourceForge JP are supported by '`shimbun`' library. Add a group name of your favorite archive to the variable `shimbun-sourceforge-jp-mailing-lists`, and you can browse it.

Elips Mailing List
: elips.elips

Squeak-ja Mailing List
: squeak-ja.main

Smalltalkers' Salon Mailing List
: sml.main

Squeak-dev Mailing List
: squeak-dev.main

Plucker Mailing List
: plucker.announce plucker.list plucker.dev

pilot-link Mailing List
: pilot-link.announce pilot-link.devel pilot-link.general pilot-link.unix-ng

Coldsync Mailing List
: coldsync.main

J-Pilot Mailing List
: jpilot.main

pilot-mailsync Mailing List
: pilot-mailsync.main

Mozilla Users Mailing List in Japan
: mozilla-jp.users

  Please note that userid and passowrd are required for '`mozilla-jp.users`' so you have to write;

```
machine www.mozilla.gr.jp
realm Please Enter mozilla mozilla
login mozilla
passwd mozilla
```

  in '`~/.w3m/passwd`' and remove group and others access permissions from the file.

tDiary Developers Mailing List in Japan
: tdiary-ml.devel tdiary-ml.theme

arch.bluegate.org Mailing Lists
: arch-bluegate.subversion-jp arch-bluegate.arch-jp arch-bluegate.mailman arch-bluegate.viewarch

Tigris.org:Open Source Software Engineering
: All archives served by Tigris.org are supported by '`shimbun`' library. Add a group name of your favorite archive to the variable `shimbun-tigris-group-alist`, and you can browse it. Group name is tigris.<project>.<mailinglist>.

www.SciPy.net Mailing Lists
: scipy.astropy scipy.ipython-user scipy.ipython-dev scipy.scipy-user scipy.scipy-dev scipy.scipy-testlog scipy.scipy-chaco scipy.scipy-cvs

9.5.4 Sport Sites Supported by Shimbun

These are sport sites supported by '`shimbun`' library.

makanai makanai.f1news

F1 FAN f1fan.news

@nifty:motorsports
msports-nifty.F1 msports-nifty.IRL msports-nifty.WRC msports-nifty.Europe msports-nifty.USA

Yahoo!SPORTS
yahoo-sports.F1 yahoo-sports.baseball yahoo-sports.keiba yahoo-sports.NBA yahoo-sports.NFL yahoo-sports.rugby

@nifty:motorsports requires the Mule-UCS package (see Section 2.3 [Other Requirements], page 3) for Emacs-21.4 or any previous versions.

9.5.5 Misc Sites Supported by Shimbun

These are misc sites supported by '`shimbun`' library. WEB BBS and serial publications are included.

Tea Cup Bulletin Boards
You can subscribe to various bulletin boards provided by Tea Cup Communication. By default, there are three pre-configured boards listed below:

tcup.yutopia
Yutopia BBS

tcup.meadow
Meadow BBS

tcup.skk SKK BBS

To add new boards to the list, look up the names and the urls and modify the `shimbun-tcup-group-alist` variable. The following form is an example to add two boards, '`foo`' and '`bar`', to the list:

```
(eval-after-load "sb-tcup"
  '(setq shimbun-tcup-group-alist
         (append
          '(("foo" "http://MMMM.teacup.com/foo/bbs2")
            ("bar" "http://NNNN.teacup.com/bar/bbs2"))
          shimbun-tcup-group-alist)))
```

2ch This is an example to browse Meadow BBS and emacs-w3m BBS on 2ch.

```
(setq shimbun-2ch-group-alist
      '(("Meadow" .
         "http://pc.2ch.net/test/read.cgi/software/1005469775")
        ("emacs-w3m" .
         "http://pc.2ch.net/test/read.cgi/unix/1013710106")))
```

Bulletin Board Systems using CGI_Board
Set your favorite browse bulletin board systems using CGI_Board to `shimbun-cgi-board-group-alist`, and you can browse them.

HNS
: This is an example to use ‘sb-hns’.

```
(setq shimbun-hns-group-alist
      '(("arisawa"                              ;; Group Name
         "http://nijino.homelinux.net/diary/" ;; URL
         "ari@mbf.sphere.ne.jp")              ;; E-Mail Address
        ("miyoshi"
         "http://www.be.wakwak.com/cgi-bin/sbox/~miyoshi/hns/"
         "miyoshi@meadowy.org")))
```

tDiary
: This is an example to use ‘sb-tdiary’.

```
(setq shimbun-tdiary-group-alist
      '(("henahena"                             ;; Group Name
         "http://www.fan.gr.jp/~ring/d/")     ;; URL
        ("yoichi"
         "http://yoichi.geiin.org/d/")))
```

Diaries at Rakuten Plaza
: This is an example to use ‘sb-rakuten’.

```
(setq shimbun-rakuten-group-alist
      '(("rakuten-id" . "email-address")))
```

EmacsWiKi
: emacswiki.changes emacswiki.diff

RSS feeds containing contents
: To use this back end, look for the RSS feeds containing contents which you would like to read, and add those groups to the `shimbun-rss-hash-group-path-alist` variable by the following way. The name of the back end is ‘rss-hash’. You may use this back end for reading mainly personal blogs.

  The parameters for each group configuration consist of the name of the group, the address of the RSS feed, the type of the mail (`t` for html), the regexp matching the start of contents, and the regexp matching the end of contents. The parameters other than the name of the group and the address of the RSS feed are optional.

  Here is an example of setting `shimbun-rss-hash-group-path-alist`. In this case, you can browse those groups as ‘rss-hash.sampleblog1’ and ‘rss-hash.sampleblog2’:

```
(setq shimbun-rss-hash-group-path-alist
      '(;; text mail
        ("sampleblog1" "http://www.example.com/index1.rss")
        ;; html mail
        ("sampleblog2" "http://www.example.com/index2.rss"
         t "</title>" "<!-- start ads")))
```

Atom feeds containing contents
: As well as the previous section (RSS feeds containing contents), you can also read the Atom feeds which contain published contents. To do that, configure the variable `shimbun-atom-hash-group-path-alist` (and possibly `shimbun-`

`atom-hash-x-face-alist`, etc.) in the way similar to shimbun-rss-hash-*. The name of the back end is '`atom-hash`'.

RSS feeds without published content

Many feeds do not contain the full content of the articles, or only so called teasers, i.e. quick summaries. If a site publishes such a feed, instead of writing a special shimbun for it, you can in many cases use the '`rss-blogs`' back end. The setup is similar to the '`rss-hash`' shimbun; here is an example:

```
(setq shimbun-rss-blogs-group-url-regexp
  '(("first-feed" "http://example/wordpressfeed")
    ("second-feed" "http://example/somefeed"
     "<div name=\"content\">" "<div name=\"comments\">")
    ("third-feed" "http://example/someotherfeed" 'none)))
```

The first two items are the name and the URL of the feed. Optionally, you can give two regular expressions denoting the start and end of the actual content on the HTML pages the feed is pointing to. If you just use the symbol `none` here, no filtering will be done whatsoever. Additionally, the '`rss-blogs`' shimbun can deal automatically with some popular blogging engines, namely Google's Blogger/Blogspot (including comment feeds), WordPress, and TypePad. If your feed is from a site using one of those (which you can see by looking at the `generator` tag), just omit the optional parameters and the code will try to extract the content automatically for you.

Wiki contents

This is an example to use '`sb-wiki`'. '`sb-wiki`' support PukiWiki and Hiki. If you don't know which regexps to set to 4th and 5th elements of an inner list, just set `nil` and you'll just see all contents of a page.

```
(setq shimbun-wiki-group-alist
      '(("pukiwiki" ;; Group Name
         "http://pukiwiki.org/index.php?cmd=rss10" ;; URL
         "webmaster@pukiwiki.org" ;; E-Mail Address
         nil ;; X-Face
         "\n<h3 id=\"" ;; regexp to represent contents start
         "</address>") ;; regexp to represent contents end
        ("hiki"
         "http://www.namaraii.com/hiki/?c=rss"
         "webmaster@fdiary.net"
         nil
         "<div class=\"section\">"
         "<div class=\"sidebar\">")
        ("apollo"
         "http://wiki.fdiary.net/apollo/?c=rss"
         "moriq@moriq.com"
         nil
         "<div class=\"section\">"
         "<div class=\"sidebar\">")
        ))
```

Yahoo! AUCTIONS

This is an example to use '`sb-yahoo-auctions`'.

```
(setq shimbun-yahoo-auctions-group-alist
      '(("mp3player" . "http://list3.auctions.yahoo.co.jp/jp/show/catleaf_rs
        ("iPod" . "http://search3.auctions.yahoo.co.jp/search_rss?p=iPod&auc
```

VineLinux Errata

vinelinux.errata.4x.i386 vinelinux.errata.4x.ppc vinelinux.errata.3x.i386 vinelinux.errata.3x.ppc vinelinux.errata.3x.alpha vinelinux.errata.2x.i386 vinelinux.errata.2x.ppc vinelinux.errata.2x.sparc vinelinux.errata.2x.alpha vinelinux.errata.1x

Japan's Cabinet Mail Magazine

kantei.m-magazine-en kantei.m-magazine-ja kantei.m-magazine-cn.hatoyama kantei.m-magazine-kr.hatoyama kantei.m-magazine-en.hatoyama kantei.m-magazine-ja.hatoyama kantei.m-magazine-en.aso kantei.m-magazine-ja.aso kantei.m-magazine-en.fukuda kantei.m-magazine-ja.fukuda kantei.m-magazine-en.abe kantei.m-magazine-ja.abe kantei.m-magazine-en.koizumi kantei.m-magazine-ja.koizumi

'`kantei.m-magazine`', '`kantei.m-magazine-cn`' and '`kantei.m-magazine-kr`' are also available for the backward compatibility.

Patent Office in Japan

jpo.news jpo.revision jpo.lawguide jpo.details

IBM developerWorks (in Japanese)

ibm-dev.autonomic ibm-dev.java ibm-dev.linux ibm-dev.opensource ibm-dev.webservices ibm-dev.xml

Pocketgames

pocketgames.news

Wincefan wincefan.news

PalmFan palmfan.news

Report of Electrical Stores Street (in Japanese)

dennou.report

PCWEB COLUMN Square

pcweb-column.jsr pcweb-column.yume pcweb-column.hreceipe pcweb-column.kita pcweb-column.shonanlife pcweb-column.kaden pcweb-column.nemurenai pcweb-column.komono pcweb-column.js pcweb-column.en pcweb-column.motherboard pcweb-column.svalley pcweb-column.architecture pcweb-column.motorlife pcweb-column.nihongoprog pcweb-column.objc pcweb-column.ide pcweb-column.music pcweb-column.itsecurity pcweb-column.soundvisual pcweb-column.osx pcweb-column.sopinion pcweb-column.ebook pcweb-column.orerobo pcweb-column.zsh pcweb-column.rikei pcweb-column.lifehack pcweb-column.world pcweb-column.guutara pcweb-column.volt pcweb-column.textclean pcweb-column.person pcweb-column.web20 pcweb-column.system

below items are also available for the backward compatibility. pcweb-column.itshihonron pcweb-column.yetanother pcweb-column.asia pcweb-column.benri pcweb-column.bytes pcweb-column.game pcweb-column.hitech pcweb-column.java pcweb-column.jisakuparts pcweb-column.scramble pcweb-column.toolexp pcweb-column.winvista pcweb-column.winxp pcweb-column.interview pcweb-column.ityougo pcweb-column.kimeuchi pcweb-column.stratesys pcweb-column.toyagain

Notes Exhibition
: lotusex.news lotusex.library lotusex.operation lotusex.primer lotusex.tips lotusex.practical lotusex.qanda lotusex.lounge lotusex.bbs

@IT forum
: atmarkit.news atmarkit.fwin2k atmarkit.fdotnet atmarkit.fsys atmarkit.fxml atmarkit.fdb atmarkit.flinux atmarkit.fnetwork atmarkit.fjava atmarkit.fsecurity atmarkit.farc atmarkit.fbiz atmarkit.fwcr atmarkit.jibun

TeX Q&A Bulletin Board
: texfaq.qanda

X51.org
: x51.anima x51.art x51.auction x51.blow x51.cabal x51.crime x51.disaster x51.edge x51.enema x51.ghost x51.homme x51.info x51.life x51.love x51.media x51.medical x51.military x51.northkorea x51.oparts x51.phallic x51.psychics x51.religion x51.science x51.top x51.ufo x51.uma x51.xfiles

eXperts Connection (eXConn)
: exconn.news

MSDN
: msdn.all msdn.netframework msdn.architecture msdn.asp.net msdn.data msdn.longhorn msdn.mobility msdn.subscriptions msdn.msdntv msdn.office msdn.security msdn.sql msdn.theshow msdn.vbasic msdn.vcsharp msdn.visualc msdn.vfoxpro msdn.vjsharp msdn.vstudio msdn.vs2005 msdn.webservices msdn.embedded msdn.xml msdn.japan.msdn msdn.japan.msdn-us

Haiku OS
: haiku-os.news haiku-os.forums haiku-os.newsletters

Foundation for a Free Information Infrastructure
: ffii.en.software-patents ffii.en.software-patents.ffii ffii.en.information-infrastructure ffii.en.project ffii.de.software-patente ffii.de.software-patente.ffii ffii.de.informations-infrastruktur ffii.fr.brevets-logiciels ffii.fr.brevets-logiciels.ffii ffii.nl.softwarepatenten ffii.nl.softwarepatenten.ffii

9.6 How to make a new shimbun module

'`Shimbun`' is a library set of emacs-w3m that enables you to read certain web contents using Gnus, Wanderlust, or Mew as if they were email messages. Here we will explain how to make a new '`shimbun`' module.

9.6.1 Overview

When you make a new '`shimbun`' module '`foobar`' for reading contents of `http://www.foobar.net`, what you have to do first is to put the following S expressions in the first part of the '`sb-foobar.el`' file:

```
(require 'shimbun)
(luna-define-class shimbun-foobar (shimbun) ())
```

We will explain what they are below, so you can understand they are just incantations now. You have to use the same suffix '`foobar`' in the file name ('`sb-foobar.el`') and the class name ('`shimbun-foobar`') as the second argument for the `luna-define-class` macro.

Major jobs of the '`shimbun-foobar`' module can be classified broadly into the following four categories (note that you may rephrase "folder" with "group" if you are a Gnus user):

1. Getting a page source from http://www.foobar.net in order to gather articles' subjects etc. when a MUA opens the '`foobar`' folder.
2. Gathering subjects and other necessary informations from the page source in order to make headlines of articles and returning them as the structured list called `headers`.
3. Getting a page source for an article from the web site, for example, http://www.foobar.net/053003.html, when MUA requires to display an article in the '`foobar`' folder, and
4. Removing cruft, e.g. advertisements, from the page source and formatting a raw article.

`shimbun-headers` of '`shimbun.el`' does the first job, `shimbun-get-headers` does the second, `shimbun-article` does the third and `shimbun-make-contents` does the last.

The `shimbun-headers` method does the first job, the `shimbun-get-headers` method does the second, the `shimbun-article` method does the third and the `shimbun-make-contents` method does the last thing. The default methods for those categories are defined in the '`shimbun.el`' module.

Open the '`shimbun.el`' file. You may see unfamiliar definitions like `luna-define-generic` or `luna-define-method` there. Hm, they look like `defun`, don't you? You may also see there's just a doc-string in the former definition and the same symbol is declared again in the later form. And further, there are some symbols only declared by the `luna-define-generic` form, not by the `luna-define-method` form. What on earth are we seeing? Isn't the program not written in the Emacs-Lisp language?

The truth is that the '`shimbun`' modules use the '`luna.el`' module provided by FLIM which enables you to write object oriented programs in the Emacs-Lisp language.

There are method programs defined rigidly for the specific purposes in the '`shimbun.el`' module. The `shimbun-headers` method gets a page source from a certain URL, the `shimbun-get-headers` method gathers subjects and other informations, etc. . . (see above). They do routine works, so they cannot take proper method to meet various web contents in the world. Eh? Oh, you shouldn't believe in a heresy!

The '`shimbun.el`' module only provides the default method functions. Remember the `defadvice` feature. There are three ways to modify the behavior of a function: `:before`, `:around` and `:after`. Similarly, each default '`shimbun`' method function can be modified for a certain purpose (note that the `:around` method-qualifier can be omitted). And it should be written specially that the modification will be effective only when the specified '`shimbun`' module is selected.

Now as you may have understood that the `luna-define-generic` form provides only a husk in a sense, the `luna-define-method` form defines an actual function which can be different for each '`shimbun`' module, and the `luna-define-class` form declares the '`shimbun`' class in the first part of the '`sb-foobar.el`' module.

9.6.2 Getting web page and header information

Let's identify a target web page URL to gather subjects and other informations first. If a web site uses a frame, a target is only one of the web pages. Second, lets create a body of the `shimbun-index-url` method function using the `luna-define-method` form in your '`sb-foobar.el`' file. And make the user customizable variable `shimbun-foobar-groups`, which we will explain later[7].

```
(defvar shimbun-foobar-url "http://www.foobar.net")

(luna-define-method shimbun-index-url ((shimbun shimbun-foobar))
  shimbun-foobar-url)

(defvar shimbun-foobar-groups '("news"))
```

After you create a body of the `shimbun-index-url` method, the `shimbun-headers` method can get a web page source since the '`shimbun.el`' module already has the default `shimbun-headers` method. After the `shimbun-headers` method gets a web page source, it calls the `shimbun-get-headers` method to gather headers information. As the '`shimbun.el`' module does not have the `shimbun-get-headers` method, you have to create it in your '`sb-foobar.el`' file.

Now look carefully in the page source and create the `shimbun-get-headers` method in your '`sb-foobar.el`' file.

Create a regular expression that can gather headers information. Minimally necessary information are subject, date, author, URL and `message-id` of an article. They are used in MUA as Subject, Date, From, Xref and Message-ID.

If you want to make an article from a line in a web page source, like:

```
<a href="053003.html">some talks on May 30(posted by Mikio &lt;foo@bar.net&gt;)</a>
```

use the following regexp:

```
"<a href=\"\\(\\([0-9][0-9][0-9][0-9]\\)[0-9][0-9]\\.html\\)\">\\([^<(]+\\)(posted by
```

You can get a value for Xref by `(match-string 1)`. You can get a value for Date by modifying a value of `(match-string 2)`. Subject by `(match-string 3)` and From from `(match-string 4)`. You can modify them further for showing additional information in MUA.

If URL of an article is a relative path like above, use `shimbun-expand-url` to expand it before putting information to header. If each article doesn't have a each unique URLs (i.e. URL of headers and URL of articles are just same), you have to ask Emacs to remember body of an article when gathering headers information, For more detail see the files '`sb-palmfan.el`', '`sb-dennou.el`' and '`sb-tcup.el`'.

Sometimes you cannot identify Date information when gathering headers information only from a web page source. If so, leave it, just set a null string, `""` to its value. If you can identify Date only when you see contents of an article, you can set it at that time by using `shimbun-make-contents` method. And you may use a fixed From for a web site (e.x. `"webmaster@foobar.net"`).

[7] At least one group is necessary for each '`shimbun`' module even if you don't want it.

Be careful when you build a message-id. Make sure it has uniqueness otherwise you may not be able to read some articles in the 'shimbun'[8]. Assure uniqueness by building message-id using date information, a domain of the page and/or a part of URL of the page. And use '@' but ':' as a part of message-id in order to display inline images. See RFC2387 and RFC822 for more detail.

Put these information to header using function shimbun-create-header of the 'shimbun.el' module.

A bare bone of shimbun-get-headers in your 'sb-foobar.el' file is as follows:

```
(luna-define-method shimbun-get-headers ((shimbun shimbun-foobar)
                                         &optional range)
  (let ((regexp "....")
        subject from date id url headers)
    ...
    (catch 'stop
      (while (re-search-forward regexp nil t nil)
        ...
        (when (shimbun-search-id shimbun id)
          (throw 'stop nil))
        (push (shimbun-create-header
               0 subject from date id "" 0 0 url)
              headers)))
    headers))
```

Note that you can access 'shimbun-foobar' instance via temporary variable shimbun in the method.

Now we will explain a user variable shimbun-foobar-groups.

Assume that you have two groups of articles in http://www.foobar.net and there are two different web pages for such groups in where 'shimbun' module gathers header information. For examples, there are what's new information of the web site in http://www.foobar.net/whatsnew/index.hmtl, and there are archive lists of email messages posted to ML in http://www.foobar.net/ml/index.html. In such case you may want to access the group by 'shimbun' folders 'foobar.whatsnew' and 'foobar.ml'. If so, put the following S expressions to the 'sb-foobar.el' file.

```
(defvar shimbun-foobar-url "http://www.foobar.net")

(defvar shimbun-foobar-group-path-alist
  '(("whatsnew" . "/whatsnew/index.html")
    ("ml" . "/ml/index.html")))

(defvar shimbun-foobar-groups
  (mapcar 'car shimbun-foobar-group-path-alist))

(luna-define-method shimbun-index-url ((shimbun shimbun-foobar))
  (concat shimbun-foobar-url
```

[8] And more, you may not be able to read actual email messages from someone when message-ids conflict!

```
(cdr (assoc (shimbun-current-group-internal shimbun)
            shimbun-foobar-group-path-alist))))
```

You can get the current group by using `shimbun-current-group-internal`. You can use it in `shimbun-get-headers` method (or others) in order to change its behavior in accordance with the current group.

Each '`shimbun`' module needs at least one group. There is not a special rule for naming a group, but if you don't find out a good name, use '`news`' or '`main`'.

9.6.3 Displaying an article

`shimbun-article` method defined in the '`shimbun.el`' module gets URL from Xref information of header, get a web page source from the URL, and call `shimbun-make-contents` in working buffer of the source. Major job of `shimbun-make-contents` is to process such HTML. Imagine that a working buffer has a web page source of an article. `shimbun-make-contents` defined in the '`shimbun.el`' module insert (i) header information to top of the buffer, (ii) '`<html>`', '`<body>`' and etc. right after the information, and (iii) '`</body>`' and '`</html>`' to end of the buffer. MUA displays an article as a HTML mail.

> Not only HTML articles, but also articles in the '`text/plain`' format can be generated. See Section 9.6.5 [Making text/plain articles], page 89.

If you don't want to process an article, you don't have to define `shimbun-make-contents` in the '`sb-foobar.el`' module.

If you want to remove some part of a web page source of an article at its top and its end, set regexp to `shimbun-foobar-content-start` that matches content start and `shimbun-foobar-content-end` that matches content end.

```
(defvar shimbun-foobar-content-start "^<body>$")
(defvar shimbun-foobar-content-end "^<\/body>$")
```

`shimbun-clear-contents`, which is called by `shimbun-make-contents` defined in the '`shimbun.el`' module, will remove HTML source from `point-min` to `shimbun-foobar-content-start` and from `shimbun-foobar-content-end` to `point-max` using the regexps. Note that it will not remove any HTML source when either of the regexp searches fails.

If you want to remove more unnecessary parts (e.x. advertisements) diligently, define `shimbun-clear-contents` in your new '`sb-foobar.el`' file as follows:

```
(luna-define-method shimbun-clear-contents :around ((shimbun shimbun-foobar)
                                                    header)
  ;; cleaning up
  (while (re-search-forward "..." nil t nil)
    (delete-region (match-beginning 0) (match-end 0)))
  (luna-call-next-method))
```

For more details see `shimbun-make-contents` in the '`sb-ibm-dev.el`' file.

I said in the subsection of Section 9.6.2 [Getting web page and header information], page 86 that if each article doesn't have a each unique URLs you have to ask Emacs to remember body of an article when gathering headers information, In such case you don't have to get a web page from URL of Xref in '`shimbun-article`' method. Just get texts from Emacs memories and put them with pretty printing. For more detail see definitions of '`shimbun-article`' method of '`sb-palmfan.el`', '`sb-dennou.el`' or '`sb-tcup.el`'.

9.6.4 Inheriting shimbun module

There are some famous mailing list manager (or archiver).

- Mailman The GNU Mailing List Manager, formerly called as '`pipermail`'. See http://www.gnu.org/software/mailman/index.html for detail.
- MHonArc See http://www.oac.uci.edu/indiv/ehood/mhonarc.html for detail.
- fml fml mailing list server/manager. See http://www.fml.org/software/fml/ for detail.
- mailarc See http://cvs.namazu.org/mailarc/ for detail.

If you find out one of such mailing list managers' names in a web page source when you analyze it in the step of See Section 9.6.2 [Getting web page and header information], page 86, you are very lucky[9]. The modules '`sb-mailman.el`', '`sb-mhonarc.el`', '`sb-fml.el`' and '`sb-mailarc.el`' have the `shimbun-get-headers` method, etc, already, when you write small code that is not defined in such '`shimbun`' modules, your new '`sb-foobar.el`' module works!

If you use the '`sb-mailman.el`' module, write the following S expressions to the top of the '`sb-foobar.el`' file:

```
(require 'sb-mailman)
(luna-define-class shimbun-foobar (shimbun-mailman) ())
```

Those above mean that '`shimbun`' module '`shimbun-foobar`' inherits shimbun-mailman class[10] and methods defined in the '`sb-mailman.el`' module will be used in '`shimbun-foobar`' by default. You can overwrite some of parent methods, if necessary.

See the '`sb-pilot-mailsync.el`' file as a sample that uses the '`sb-mailman.el`' module. You can feel how easy to create a new '`shimbun`' module by using such parent modules.

Note that there are some localized version of such mailing list manager, for examples, some of them show Date information in Japanese. The modules '`sb-mailman.el`', '`sb-mhonarc.el`', '`sb-fml.el`' and '`sb-mailarc.el`' assumes that mailing list managers are not localized.

If you want to read via '`shimbun`' a web site that uses localized mailing list manager, you may have to overwrite some methods in the parent module.

9.6.5 Making text/plain articles

Even if the MUA is reinforced by emacs-w3m so as to be able to read HTML articles, '`text/plain`' articles might be more convenient in some cases. To make the '`sb-foobar`' module generate '`text/plain`' articles rather than '`text/html`' articles, there are two ways to do that.

- The one is to make the '`sb-foobar`' module inherit (see Section 9.6.4 [Inheriting shimbun module], page 89) the '`sb-text`' module. Here's an example you may put in the beginning of the '`sb-foobar`' module.

```
(require 'sb-text)
(luna-define-class shimbun-foobar (shimbun-text) ())
```

[9] Such mailing list managers often show their own name in an archive list page

[10] i.e. shimbun-mailman class is a parent class.

The '`sb-text`' module provides the `shimbun-make-contents` method which generates the articles in the '`text/plain`' format. This will be useful for the '`shimbun`' modules handling the web sites which put up only text articles.

- The other is to set the `shimbun-foobar-prefer-text-plain` variable to non-`nil`. This makes the `shimbun-make-contents` method generate the articles in the '`text/plain`' format (actually, it uses the functions provided by the '`sb-text`' module). Note that this is effective only to the modules which inherit the default `shimbun-make-contents` method (especially the modules which inherit the '`sb-text`' module are not affected). The advantage of this way is that users can easily switch '`text/plain`' articles and '`text/html`' articles.

  The default value for the `shimbun-foobar-prefer-text-plain` variable is `nil` if it is not defined. So, it defaults to `nil` in every '`shimbun`' module except for the modules '`sb-asahi.el`' and '`sb-yomiuri.el`'.

  In addition, you can use the variables `shimbun-foobar-text-content-start` and `shimbun-foobar-text-content-end` instead of `shimbun-foobar-content-start` and `shimbun-foobar-content-end` to extract significant text in web pages (see Section 9.6.3 [Displaying an article], page 88). If the formers are not defined, those values default to the latter values.

Whichever the ways you use, you should note that the '`text/plain`' articles cannot contain images, links, etc.

9.6.6 Zenkaku to hankaku conversion

"Zenkaku" or "zenkaku character(s)" is a term commonly used to call Japanese wide characters, and "hankaku" is an opposite term for ordinary ASCII characters. There is a complete set of zenkaku characters corresponding to at least the ASCII character set.

Some Japanese web sites tend to use zenkaku characters a lot, and those articles might not necessarily be comfortable to read. If you feel so, you can use this feature that converts those zenkaku ASCII characters into hankaku. To do that, set the `shimbun-foobar-japanese-hankaku` variable to `t`. Where `foobar` is a server name to which you subscribe for shimbun articles. That is, you have to use it per server.

If you prefer to convert zenkaku to hankaku only in the body of articles, use the value `body` instead of `t`. Contrarily the value `header` or `subject` specifies to perform it only in subjects.

9.6.7 Coding convention of Shimbun

- You can use all functions defined in emacs-w3m in '`shimbun.el`'.
- You can use no functions defined in emacs-w3m in '`sb-*.el`'. If you want to use emacs-w3m functions in '`sb-*.el`', you must add their stubs to '`shimbun.el`'.
- You must avoid file names that have already used in SpeedBar. Here is a list of file names used in '`speedbar-0.14beta4`'.

sb-ant.el	sb-html.el	sb-info.el	sb-texinfo.el
sb-gud.el	sb-image.el	sb-rmail.el	sb-w3.el

- You should select file names which remind their referring WEB servers' URIs. It is allowed to remove country parts (Ex. jp,de,uk,etc), organization parts (Ex.

edu,com,org,net,etc) and redundant parts (Ex. www) if removing does not increase vagueness.

- You should select group names in USENET style. It means that small characters are preferred in group names, and that period(.) is preferred as an delimiter to show hierarchical structure in groups.

10 Some knick-knacks using emacs-w3m

Here are some handy tips to use emacs-w3m with other Emacs facilities.

- browse-url

  You can use emacs-w3m with the `browse-url` feature. For instance, put the following lines in your '`~/.emacs`' file:

  ```
  (setq browse-url-browser-function 'w3m-browse-url)
  (global-set-key "\C-xm" 'browse-url-at-point)
  ```

  Emacs-w3m will now be invoked when you type the *C-x m* key on a string which looks like a URL in any Emacs buffer. In addition, you can use emacs-w3m to preview an HTML file that you are just editing by typing the *C-c C-v* key (note that you need to use Emacs and the `html-mode` major mode to edit the HTML file).

  If you'd like to use another web browser than emacs-w3m when using the *C-x m* key when you are in an emacs-w3m buffer (who wants to do so?), add the following advice to '`~/.emacs`' file:

  ```
  (defadvice browse-url-at-point
    (around change-browse-url-browser-function activate)
    "Use Netscape only when it is invoked in an emacs-w3m buffer."
    (let ((browse-url-browser-function
           (if (eq major-mode 'w3m-mode)
               'browse-url-netscape
             'w3m-browse-url)))
      ad-do-it))
  ```

- dired

  You can use emacs-w3m to browse an HTML file in a `dired` buffer by typing the *C-x m* key. Use the following settings in your '`~/.emacs`' file:

  ```
  (eval-after-load "dired"
    '(define-key dired-mode-map "\C-xm" 'dired-w3m-find-file))

  (defun dired-w3m-find-file ()
    (interactive)
    (require 'w3m)
    (let ((file (dired-get-filename)))
      (if (y-or-n-p (format "Use emacs-w3m to browse %s? "
                            (file-name-nondirectory file)))
          (w3m-find-file file))))
  ```

- hnf-mode

  You can see the newest diary using emacs-w3m and the hyper nikki system. Put the following lines in your '`~/.emacs`' file and type the *C-c C-b* key in an `hnf-mode` buffer:

  ```
  (defun w3m-hnf-browse-url-w3m (url &optional new-window)
    (interactive (browse-url-interactive-arg "URL: "))
    (save-selected-window
      (pop-to-buffer (get-buffer-create "*w3m*"))
      (w3m-browse-url url new-window)))
  (setq hnf-browse-url-browser-function #'w3m-hnf-browse-url-w3m)
  ```

- Gnus

  You've mistaken the entrance if you are a Gnus user and this section is the first page you read in this Info. See Chapter 6 [Hooking into MUAs], page 49 first.

  By default, Gnus will not apply the treatment variables, for instance `gnus-treat-strip-banner`, to '`text/html`' parts. To have them applied to '`text/html`' parts automatically, there are two ways to do that:

```
;; Apply all the treatments to text/html parts.
(eval-after-load "gnus-art"
  '(add-to-list 'gnus-article-treat-types "text/html"))

;; Apply a certain treatment to text/html parts.
(setq gnus-treat-strip-banner '(or t (typep "text/html")))
```

  Also See section "Customizing Articles" in *The Gnus Manual*, for details.

  In addition, the experimental code below is used to display '`multipart/related`' pictures. The place might be something wrong.

```
(eval-after-load "gnus-art"
  '(or (assoc "multipart/related" gnus-mime-multipart-functions)
       (setq gnus-mime-multipart-functions
             (cons
              (cons
               "multipart/related"
               (byte-compile
                (lambda (handle)
                  (gnus-mime-display-mixed (cdr handle)))))
              gnus-mime-multipart-functions))))
```

- yahtml-mode

  You can use emacs-w3m to preview an HTML file that just you are editing with the `yahtml-mode`. Here is an example:

```
(defadvice yahtml-browse-html
  (around w3m-yahtml-browse-html activate compile)
  (w3m-goto-url (ad-get-arg 0) t))
```

- jisx0213

  You can use JIS X 0213 character set in Emacs using the `jis0213` module which is bundled in the Mule-UCS package. Although the `decode-char` function is overridden by `mucs` (`jis0213` loads `mucs`) and it stops working properly for the `ucs` coded character set, it has been reported that also to load the `unicode` module seems to solve the problem. The reason has not been made clear yet. Here is an example for the '`~/.emacs`' file:

```
(require 'unicode)
(require 'un-define)
(require 'jisx0213)
```

11 Mailing list and submitting bug reports

We have set up a mailing list to discuss all things emacs-w3m. You can post without subscribing. If you find a bug, have a feature request, or have written some code, don't hesitate to post to the list. And if you're just a user and like the program, please tell us too!

The address is:

Emacs-w3m Mailing List `<emacs-w3m@namazu.org>`

You can also send a bug report using the `report-emacs-w3m-bug` command (or the *C-c C-b* key) if you have set the `mail-user-agent` variable that will work properly.

English and Japanese can be used when posting to this list, since many of its members are Japanese. Articles posted to the list are opened to the public and you can read them on the web (at http://emacs-w3m.namazu.org/ml/), or in NetNews (group '`gmane.emacs.w3m`' on the server '`news.gmane.org`').

If you want to receive articles by mail, send a mail containing

```
subscribe Your Name
```

(please write your name, not your email address) in its body to '`emacs-w3m-ctl@namazu.org`', then you can subscribe to the list. To unsubscribe from it, send a mail containing just

```
# bye
```

in its body to '`emacs-w3m-ctl@namazu.org`'.

12 Details of some emacs-w3m functions

13 Companion packages you might need

Even though emacs-w3m provides a wealth of features, you may want to check out the following external packages for even more usability:

- w3m-type-ahead.el

  The w3m-type-ahead.el package provides "type ahead" searching, similar to the feature by the same name from Mozilla browsers. It allows you to find anchors using an interface similar to isearch, but results are limited to anchors in the buffer.

  Download w3m-type-ahead.el from http://alioth.debian.org/project/showfiles.php?group_id=30594.

- newsticker.el

  newsticker.el that has been incorporated in Emacs 22 and greater is a rowse rss feeds and also atom feeds. Here is a configuration example to use newsticker.el together with emacs-w3m (put it in the '`~/.emacs`' file):

  ```
  (require 'w3m-load)
  (setq newsticker-html-renderer 'w3m-region
        browse-url-browser-function 'w3m-browse-url)
  ```

  see section "Top" in *A Newsticker for Emacs*, for details.

14 People who wrote this manual

- Romain Francoise
- NAKAJIMA Mikio
- Yoichi NAKAYAMA
- Ryoko NARITOKU(Translation only)
- Hideyuki SHIRAI
- TSUCHIYA Masatoshi
- Katsumi Yamaoka
- Masatake YAMATO
- Naohiro Aota

Index

Concept Index

T

U

V

W

Z

Key Index

+

,

.

<

=

>

[

]

^

{

}

\

|

A

B

C

D

E

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

Y

Variable Index

G

M

N

S

W

Function Index

G

M

N

O

W