Lisp-history-9.1 インストール・マニュアル

まず、line-edit-9.1.tar.gz を展開するディレクトリを決めます。

ここではホーム・ディレクトリの下の lisp ディレクトリの下に「line-edit-9.1」というディレクトリに展開することにします。初期設定用ファイルの 2 行目で、このディレクトリ名を設定していますので、他のディレクトリに展開する場合は初期設定ファイルの 2 行目のディレクトリ名も書き換えてください。

・line-edit-9.1.tar.gz をインストールしようとするユーザ名が daigo
・ダウンロードした line-edit-9.1.tar.gz が /home/daigo/download ディレクトリにある。
・GNU Common Lisp 2.5.3 が /home/daigo/lisp/gcl-2.5.3 以下にある。

以上の条件であるならば、 インストール手順は以下のようになります。

```
[1062]> cd /home/daigo/lisp
[1063]> mkdir line-edit-9.1
[1064]> cp /home/daigo/download/line-edit-9.1.tar.gz line-edit-9.1/
[1065]> cd line-edit-9.1
[1066]> tar zxf line-edit-9.1.tar.gz
[1067]> cp top.lsp /home/daigo/lisp/gcl-2.5.3/lsp
[1068]> cd ../gcl-2.5.3
[1069]> ./configure
[1070]> make
[1071]> make install
[1072]> vi /usr/local/bin/gcl
[1073]> cd ../line-edit-9.1
[1074]> cp .gclrc.lsp /home/daigo/
[1075]> ./init-line-edit
```

[1072]では gcl 起動用シェル・スクリプト内の readline ライブラリ初期化部分の削除を行います。GCL 2.5.x がサポートする readline ライブラリと line-edit が競合するためです。readline ライブラリに比べ line-edit を使用した方が、Lisp セッション独自に履歴を管理できたり、S 式単位での編集が行え

るなど高機能です。readline ライブラリから削除するのは以下の行です。

```
-- /usr/local/bin/gcl
#!/bin/sh
export C_INCLUDE_PATH=/usr/local/lib/gcl-2.5.3/h:$C_INCLUDE_PATH
exec /usr/local/lib/gcl-2.5.3/unixport/saved_gcl ¥
    -dir /usr/local/lib/gcl-2.5.3/unixport/ ¥
    -libdir /usr/local/lib/gcl-2.5.3/ ¥
    -eval '(setq si::*allow-gzipped-file* t)' ¥
    -eval '(si::init-readline)' ¥                    <==    この行を削除
      "$@"
# other options: -load /tmp/foo.o -load jo.lsp -eval "(joe 3)"
-- end /usr/local/bin/gcl
```

GCL 起動用シェルスクリプトの gcl ファイルが存在するパスや内容は GCL を make する際の configure スクリプトの実行結果に依存するので異なる部分があるかも知れません。また、このスクリプト・ファイルは GCL を make する毎に新規に作成されるのでご注意下さい（テンプレートは makefile の 134 行目にあります)。

以上でインストールは終了です。[1067]から[1071]は top.lsp を入れ替えた GCL の make です。

Alias 定義

これで、以後の実行は

```
[1076]> /usr/local/bin/gcl -load /home/daigo/.gclrc.lsp
```

で line-edit-9.1 の初期設定が行われた状態で GCL が起動します。alias 定義しておくと便利でしょう。

```
[1077]> alias gcl='/usr/local/bin/gcl -load /home/daigo/.gclrc.lsp'
```

これで、以後は

```
[1078]> gcl
GCL (GNU Common Lisp)  (2.5.3) Mon May 12 17:30:14 JST 2003
Licensed under GNU Library General Public License
Dedicated to the memory of W. Schelter

Use (help) to get some basic information on how to use GCL.
[0]>
```

となります。

history-pkg.lsp と line-edit-pkg.lsp を連動させて使用する場合は依存関係があるので history-pkg.lsp、line-edit-pkg.lsp の順にロードする必要があります。コンパイルする場合もこの順で行う必要があります。更に、パッケージ名の衝突を避けるために、line-edit-9.1 をロードしていない状態でコンパイルしてください。

個別に使用する場合はロード、コンパイルの順序に制限はありません。

個別に使用する場合（C-p, C-n は動作しない）

```
[1079]> /usr/local/bin/gcl
GCL (GNU Common Lisp)  (2.5.3) Mon May 12 17:30:14 JST 2003
Licensed under GNU Library General Public License
Dedicated to the memory of W. Schelter

Use (help) to get some basic information on how to use GCL.
> (push 'disable-hook *features*)
 (disable-hook :compiler :numlib :sdebug :defpackage truncate_use_c
        clx-little-endian bsd mc68020 bsd386 sgc ieee-floating-point unix
        gmp broken_o4_opt gcl akcl common kcl)
> (compile-file "history-pkg.lsp")
```

Compiling history-pkg.lsp.
End of Pass 1.
End of Pass 2.
OPTIMIZE levels: Safety=0 (No runtime error checking), Space=0, Speed=3
Finished compiling history-pkg.lsp.
#p"history-pkg.o"
> (compile-file "line-edit-pkg.lsp")

Compiling line-edit-pkg.lsp.
End of Pass 1.

;; Note: Tail-recursive call of GET-COMMAND was replaced by iteration.
;; Note: Tail-recursive call of TRUE-GET-COMMAND was replaced by iteration.
;; Note: Tail-recursive call of CONDENSE was replaced by iteration.
;; Note: Tail-recursive call of CONDENSE was replaced by iteration.
;; Note: Tail-recursive call of CONDENSE was replaced by iteration.
;; Note: Tail-recursive call of MORE-SIMPLE-CONDENSE was replaced by iteration.
End of Pass 2.
OPTIMIZE levels: Safety=0 (No runtime error checking), Space=0, Speed=3
Finished compiling line-edit-pkg.lsp.
#p"line-edit-pkg.o"
>

連動して使用する場合（推奨設定です。C-p, C-n も動作する）

[1079]> /usr/local/bin/gcl
GCL (GNU Common Lisp) (2.5.3) Mon May 12 17:30:14 JST 2003
Licensed under GNU Library General Public License

Dedicated to the memory of W. Schelter

Use (help) to get some basic information on how to use GCL.
> (compile-file "history-pkg.lsp")

Compiling history-pkg.lsp.
End of Pass 1.
End of Pass 2.
OPTIMIZE levels: Safety=0 (No runtime error checking), Space=0, Speed=3
Finished compiling history-pkg.lsp.
 #p"history-pkg.o"
> (compile-file "line-edit-pkg.lsp")

Compiling line-edit-pkg.lsp.
End of Pass 1.

;; Note: Tail-recursive call of GET-COMMAND was replaced by iteration.
;; Note: Tail-recursive call of TRUE-GET-COMMAND was replaced by iteration.
;; Note: Tail-recursive call of CONDENSE was replaced by iteration.
;; Note: Tail-recursive call of CONDENSE was replaced by iteration.
;; Note: Tail-recursive call of CONDENSE was replaced by iteration.
;; Note: Tail-recursive call of MORE-SIMPLE-CONDENSE was replaced by iteration.
End of Pass 2.
OPTIMIZE levels: Safety=0 (No runtime error checking), Space=0, Speed=3
Finished compiling line-edit-pkg.lsp.
 #p"line-edit-pkg.o"
>

起動時実行ファイルのサンプル

gcl コマンドの起動時実行ファイル（init.lsp、または -load で指定するファイル）のサンプルと、history-pkg、line-edit-pkg 用の各設定ファイル history.pkg, line-edit.pkg の例を示します。以下の例ではホーム・ディレクトリ下の lisp ディレクトリの下の line-edit-9.1 に各種設定ファイルと history-pkg.{o,lsp}、line-edit-pkg.{o,lsp}を置くと仮定しています。

```
---- 初期設定用ファイル .gclrc.lsp のサンプル
(defconstant *homedir* (namestring (user-homedir-pathname)))
(defconstant *default-load-path* (concatenate 'string *homedir* "lisp/line-edit-9.1/"))

(setf *load-verbose* nil)
;(setf *load-verbose* t) (si::use-fast-links nil)

(setf *print-case* :downcase)

;
; load 'history-pkg'
;
(load (merge-pathnames "history.pkg" *default-load-path*))

;
; load 'line-edit-pkg'
;
(load (merge-pathnames "line-edit.pkg" *default-load-path*))
(set-editor-mode "emacs-mode" *default-load-path*)

;
; hook for GCL/KCL
;       save histories automaticaly when exit.
;       save all registers automaticaly when exit.
;
(shadow 'bye)
```

```
(defun bye (&optional (status 0))
  (write-history)
  (save-edited-fname *default-load-path*)
  (save-registers)
  (lisp:bye status))

;;
;; コントロール・キーやメタ・キーの入力を即座に処理するためにはバッファリ
;; ングなし、入力文字のエコーなしの入力モード（raw-mode）が必須である。し
;; かし、raw-mode での処理中に何らかのエラーが発生して、cooked-mode に設定
;; する関数(cooked-mode)に制御が渡らなくなると、入力文字がエコー・バック
;; されない。
;;
;; エラーが発生した場合には関数 error が呼ばれる。そこで cooked-mode に戻して
;; から、本来の関数に制御を渡すように error を再定義する。
;;
(shadow 'error)

(defun error (fmt &rest args)
  (cooked-mode)
  (eval (cons 'lisp:error (cons fmt args))))

;;
;; for line-edit-pkg and history-pkg.
;;
(defun system:top-level-read ()
  (let (val str (eos (cons nil nil)))
    (loop
      (setf str (line-edit))
      (setf val (with-input-from-string (stream str) (read stream nil eos)))
```

```
        (add-to-str-hist (format nil "~s" val))
        (when (not (eq val eos)) (return val)))))
---- <EOF>

---- history.pkg
(require 'history-pkg (merge-pathnames "history-pkg.o" *default-load-path*))
(use-package 'history-pkg)
(history-file (merge-pathnames (history-file) *default-load-path*))
(history-buffer (merge-pathnames (history-buffer) *default-load-path*))
(restore-history)                ; If you like continued history number.
                                 ; May use (read-history) also.
(last-histories 40)              ; set default number of prints.
---- <EOF>

---- line-edit.pkg
(require 'line-edit-pkg (merge-pathnames "line-edit-pkg" *default-load-path*))
(use-package 'line-edit-pkg)

(registers-file (merge-pathnames ".gcl.registers" *default-load-path*))

(restore-registers)
(load-macro (macro-file (merge-pathnames (macro-file) *default-load-path*)))

(kill-ring-max 30)
(blink-paren-deley 3/10)
(blink-paren-just-inserted t)
(undo-limit nil)
(line-size 72)
(auto-scroll-offset t)
(newline-symbol #¥$)
(kbd-macro-query-time 1)
(enable-audio-bell t)
---- <EOF>
```