Getting Started in Gnu Common Lisp

Some system stuff for our Gnu Common Lisp

1. Enter GCL by typing (from the Unix prompt)
   gcl
   NOTE: The GCL prompt is: >

2. Exit GCL by typing (from the Lisp prompt)
   (bye) (or: (by)) Note: the ( ) are part of the Lisp exit command.

3. NOTE: An error in the Lisp environment changes the prompt to: >>
   To return to the top-level of Lisp after an error type: :q
   Note: the : is part of the return to top-level command.

4. To load a file (which is in the direcory you were in when you
   got into Lisp) into the Lisp environment:
   (load "filename.ext")
   NOTE: GCL converts everything not inside double quotes to uppercase.
   So (load 'filename.ext) will look for (and work for) FILENAME.EXT.

5. To edit a file from within the Lisp environment: (ed "filename.ext")
   This command will pop you into the vi editor. When you exit the vi
   editor, you will be returned to the Lisp environment. You may also
   use ctrl-z from within the Lisp environment to pop out to the Unix
   command line. A fg command will return you to the Lisp enviornment
   you left. As noted above, (ed 'filename.ext) will look for/create
   a file named FILENAME.EXT.

6. (dribble 'filename.ext) at the GCL prompt will start a log file from
   within the GCL environment.
   (dribble) will stop logging and close the log file. Note: the name
   of the file will be FILENAME.EXT. If you use (dribble "filename.ext")
   then the name of the file will be filename.ext. You can also start
   a script file before entering the Lisp environment just like we did
   with Prolog.

7. (trace fn1name fn2name ... fniname) will cause subsequent calls to
   those functions to be traced.
   (untrace fn1name fn2name ... fniname) will turn off the trace.

8. Lisp syntax is not case-sensitive (except inside strings which are
   delimited with "string contents"). I.e., BAT, bat, Bat are all the
   same variable name.

Resources, documentation, sources of information

1. Documentation is available on the web.
   the URL's of documentation and tutorial sites.

2. Within the Lisp environment, some information about builtin functions

is available if you know the name of the function via describe. E.g., to see information about the builtin car use: (describe 'car) If there is a lot of information, it will scroll off the screen, even beyond the reach of the terminal window scroll bar. If you want to capture the information, dribble it to a file.

3. If you do not know the exact name of the function but know or can guess part of it, use (apropos "partial-name") to obtain a list of names containing the partial-name.

Notes on Lisp programs

1. A ; indicates that the rest of the line is a Lisp comment
   A commonly adopted style uses ;;; for major headings, ;; for function headings and ; for line comments.

2. Since matching parenthesis is so important in Lisp (and unmatched parentheses the most common error) it is good to have an easy way to make sure parentheses are matched. Within the vi editor, if the cursor is over a parenthesis, typing % will move the cursor to the matching parenthesis. If the cursor does not move, there is no matching paren.
   Also, within vi, you can type :set sm
   Then whenever you type a) the cursor momentarily jumps to the matching (before returning to the spot it left.

3. Although it is not syntactically necessary, good Lisp programming dictates using multiple lines and indentation to make s-expressions more readable.