

```

1 ; vim: ts=2 et sw=2 et:
2 ;;; macros -----
3 (defmacro if (test yes &optional no)
4   "Anaphoric if (traps result of conditional in 'it')."
5   `(let ((it ,test)) (if it ,yes ,no)))
6
7 (defmacro while (expr &body body)
8   "Anaphoric while (traps result of conditional in 'a')."
9   `(do ((a ,expr ,expr)) ((not a)) ,@body))
10
11 (defmacro $ (x &optional (our *config*))
12   "Access a config variable name."
13   `(fourth (assoc ',x (our-options ,our))))
14
15 (defmacro ? (s x &rest xs)
16   "Nested access to slots."
17   (if (null xs) `(slot-value ,s ',x) `(? (slot-value ,s ',x) ,@xs)))
18
19 (defmacro with-csv ((lst file &optional out) &body body)
20   "File row iterator."
21   `(progn (csv ,file #'(lambda (,lst) ,@body)) ,out))
22
23 ;;; random -----
24 (defun randf (&optional (n 1.0))
25   (setf ($ seed) (mod (* 16807.0d0 ($ seed)) 2147483647.0d0))
26   (* n (- 1.0d0 (/ ($ seed) 2147483647.0d0))))
27
28 (defun randi (&optional (n 1))
29   (floor (* n (/ (randf 1000000.0) 1000000))))
30
31 ;;; strings -----
32 (defun trim (x)
33   "Remove whitespace front and back."
34   (string-trim '(#\Space #\Newline #\Tab) x))
35
36 (defun num? (x)
37   "Return a number, if you can. Else return trimmed string."
38   (let ((y (ignore-errors (read-from-string x))))
39     (if (numberp y) y (let ((x (trim x)))
40       (if (equal x "") #\? x)))))
41
42 (defun subseqs (s &optional (sep #\,) (n 0))
43   "Separate string on 'sep'."
44   (aif (position sep s :start n)
45     (cons (subseq s n it) (subseqs s sep (1+ it)))
46     (list (subseq s n))))
47
48 ;;; operating system -----
49 (defun args ()
50   "Return list of command line arguments."
51   #+clisp (cdddr (cddr (coerce (EXT:ARGV) 'list)))
52   #+sbcl (cdr sb-ext:*posix-argv*))
53
54 (defun csv (file &optional (fn #'print))
55   "Send to 'fn' one list from each line."
56   (with-open-file (str file)
57     (loop (funcall fn
58       (subseqs (or (read-line str nil) (return-from csv)))))))
59
60 (defun cli (&optional (our (make-our)) (lst (args)))
61   "Maybe update 'our' with data from command line."
62   (labels ((cli1 (flag x) (aif (second (member flag lst :test #'equalp))
63     (cond ((equal x t) nil) ; flip boolean
64           ((equal x nil) t) ; flip boolean
65           (t (or (num? it) x)))
66     x)))
67   (dolist (x (our-options our) our)
68     (setf (fourth x) (cli1 (second x) (fourth x)))))

```